THÈSE

présentée au
Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS
en vue d'obtenir le titre de

Docteur de l'Institut National des Sciences Appliquées de Toulouse

Spécialité : Informatique

par

Carlos AGUILAR MELCHOR

Ingénieur de l'École Polytechnique

Les communications anonymes à faible latence

Jury :

M. J. Stern	Président	M. Y. Deswarte	Directeur de thèse
M. A. Gabillon	Rapporteur	M. J.M. Couveignes	Examinateur
M. J.J. Quisquater	Rapporteur	M. N. Sendrier	Examinateur

Cette thèse a été préparée dans le cadre du
projet PRIME, Privacy and Identity Management for Europe.

Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS
7 avenue du Colonel Roche, 31077 Toulouse CEDEX 4

À mon frère, qui est resté près de mes parents à s'occuper d'eux ces nombreuses années pendant que moi je vivais mes rêves. À mes parents, qui ont accepté de m'avoir loin d'eux malgré le chagrin que cela leur a causé.

Avant-Propos

Les travaux présentés dans ce mémoire ont été effectués au Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS) du Centre National de la Recherche Scientifique (CNRS). Je remercie dans un premier temps Jean-Claude Laprie, ancien directeur du laboratoire, puis dans un deuxième temps son successeur Malik Ghallab, pour m'avoir permis de réaliser mes travaux au LAAS.

Je tiens à remercier fortement David Powell, ancien responsable du groupe TSF, et son successeur Jean Arlat, pour m'avoir accueilli dans ce groupe. Leur confiance m'a permis de réaliser un grand nombre de missions et ils ont toujours su être encourageants et avant tout humains dans la direction de ce groupe.

J'exprime ma très vive reconnaissance à Yves Deswarte, Directeur de Recherche CNRS, qui a encadré mes travaux de thèse. Il m'a donné toute la liberté que je pouvais souhaiter tout en étant présent à chaque fois que je lui ai demandé son aide. Son appui m'a permis de réaliser des échanges enrichissants avec de nombreux chercheurs en France et à l'étranger. Ses commentaires ont dévoilé des failles importantes dans mes travaux et aidé à trouver des solutions à celles-ci. Je suis ravi d'avoir fait sa connaissance et j'espère que nous continuerons à travailler ensemble pendant de nombreuses années.

J'exprime ma gratitude à Jaques Stern, directeur du Laboratoire d'Informatique de l'Ecole Normale Supérieure, pour tout d'abord m'avoir recommandé auprès de Yves pour réaliser mon stage de fin de DEA au LAAS, puis pour m'avoir fair l'honneur de présider mon jury de thèse malgré ses importantes responsabilités.

Je remercie Alban Gabillon, professeur à l'université de Pau, ainsi que Jean-Jacques Quisquater, professeur à l'université de Louvain (à Louvain-la-Neuve, Belgique), pour avoir accepté d'être rapporteurs de ma thèse. Je remercie également Jean-Jacques Quisquater pour m'avoir accueilli pour un séjour d'un mois à son université. Ce séjour m'a permis d'avancer significativement la rédaction de ce manuscrit ainsi que de rédiger plusieurs articles et de rencontrer plusieurs doctorants et chercheurs avec qui des collaborations sont maintenant en cours.

Je remercie également Nicolas Sendrier, responsable du projet CODES à l'INRIA Rocquencourt, et Jean-Marc Couveignes, professeur à l'université Toulouse II, d'avoir accepté de faire partie de mon jury.

Je remercie sincèrement Nicolas Sendrier ainsi que les autres membres du projet CODES, de m'avoir fait découvrir les joies de la recherche lors de mon stage de fin d'études d'ingénieur. Les discussions amicales autour d'un café au sein de ce projet restent pour moi un modèle de convivialité et de camaraderie au sein d'un groupe de recherche.

Je remercie aussi Jean-Marc Couveignes de m'avoir donné son temps de façon désintéressée dans les balbutiements de ma recherche. Les après-midis au Mirail à discuter sur les courbes elliptiques et les systèmes homorphiques confirmèrent ma vocation de chercheur et m'encouragèrent a réaliser un maximum de rencontres et d'échanges avec d'autres chercheurs.

Je n'oublie pas non plus Fabrice Frances, professeur à l'ENSICA et responsable de l'enseignement en Informatique. Je le remercie pour la confiance qu'il m'a attribuée en m'assignant la responsabilité intégrale de plusieurs cours en à peine deux ans alors que je n'était qu'un simple doctorant vacataire. Je le remercie aussi pour l'accueil amical qu'il m'a donné à chaque fois que je suis allé le voir dans son bureau et pour m'avoir permis de confirmer ma vocation d'enseignant.

Je remercie très chaleureusement les membres du groupe TSF. Les membres permanents, qui débordent d'humour et de générosité, vouent une grande partie de leur temps à s'assurer que les doctorants aient tout ce dont ils ont besoin pour travailler, à leur assurer un futur emploi et dépassent très très souvent leur rôle d'encadrant purement scientifique et administratif. Les doctorants, qui assurent une permanente bonne ambiance dans le groupe, ont égayé mes journées de travail et j'espère que les liens d'amitié que j'ai tissé avec eux trouveront une continuation autant au niveau professionnel que personnel.

Je remercie sincèrement Ghislaine Chortey pour les illustrations du premier chapitre, ainsi que les gens qui ont relu ce manuscrit.

Finalement, je tiens à exprimer ma profonde gratitude à la famille Dorbes : à Philippe, bien sûr, ami aussi proche qu'on puisse l'être, à Françoise, Pierre, Cécile et Laurent, qui m'ont visité quand j'étais hospitalisé, acueilli dans leur foyer quand j'étais convalescent, et avec qui j'ai partagé de nombreux repas de famille. J'espère pouvoir leur rendre un jour tout ce qu'ils ont fait pour moi.

Table des matières

Introduction générale

Parmi les différents domaines de la protection de la vie privée dans les réseaux informatiques [DA06a, DA06c, DA06b], l'anonymisation des communications est un des axes donnant lieu au plus grand nombre de publications chaque année. Depuis quelques années, en particulier depuis la généralisation des connexions Internet à haut débit, les échanges d'informations entre ordinateurs se sont multipliés. Dans certains cas, comme lors de l'envoi ou la réception d'un courriel, cet échange est unidirectionnel ou bidirectionnel avec une intéractivité très basse (entre un premier courriel et sa réponse s'écoulent au minimum quelques minutes). Cependant, de nos jours, la plupart de ces échanges d'information se font de façon bidirectionnelle et interactive. C'est bien évidemment le cas pour tout service utilisant le protocole TCP/IP comme par exemple le transfert de fichiers par FTP, ou la navigation Web, mais c'est aussi le cas, indépendamment du protocole utilisé, pour des services qui de façon inhérente nécessitent des échanges de messages avec une faible latence, comme par exemple pour les messageries instantanées ou la voix sur IP.

Ces échanges s'opèrent généralement au travers de canaux de communication non sécurisés et donc il faut prendre en compte la possibilité qu'il y ait des écoutes, voire des modifications. Pour le contenu des messages échangés, il est possible d'utiliser des systèmes de chiffrement et de signature pour se prémunir contre ces attaques. Dans le cas des méta-informations propres aux communications, on peut aussi souhaiter qu'elles soient occultées. Ainsi, dans certains cas, la durée, le débit, les interlocuteurs, ou même l'existence d'une communication devront être considérées comme des données confidentielles. Savoir qu'une personne communique, par exemple, à partir de son ordinateur à son domicile est une information qui, à elle seule, peut donner des indications importantes sur la vie privée de l'individu en question, comme par exemple quand il est chez lui, ou quand il est réveillé. On peut imaginer des scénarios où ceci pose un problème sérieux. Par exemple, des voleurs peuvent essayer de repérer des individus ne communiquant

pas à travers leur connexion Internet pendant une longue période et ainsi obtenir des indices sur quels sont les domiciles où des familles sont en vacances.

On dispose de plusieurs primitives permettant d'anonymiser les communications (c'est-à-dire occulter les méta-informations associées) : l'utilisation de relais, la diffusion avec adressage implicite, l'émission de bourrage chiffré, et l'émission par envoi superposé. Cependant, malgré la diversité de ces primitives, tous les services qui de nos jours proposent une anonymisation des communications sont basés uniquement sur l'utilisation de relais. Plusieurs raisons motivent ce choix. En effet, les relais :

– permettent d'interconnecter des dizaines ou des centaines de milliers d'utilisateurs,
– sont bien adaptés aux réseaux à grande échelle,
– restent performants même si la fréquence des connexions et déconnexions est importante.

Pourtant, actuellement, tous les services d'anonymisation des communications ont des performances qui sont très inférieures à celles que l'on peut obtenir sans anonymiser les communications, en particulier pour ce qui concerne la latence. Ainsi, lorsqu'on utilise un service d'anonymisation des communications, le délai d'affichage des pages Web est très irrégulier, et il n'est pas possible de réaliser des communications hautement interactives, par exemple lorsqu'on utilise un système de voix sur IP.

Dans ce mémoire, nous étudierons comment obtenir des systèmes fournissant un service d'anonymisation des communications à faible latence. À ces fins, nous incluons certaines contraintes dans les groupes d'utilisateurs, comme :

– être de petite taille,
– être composé d'utilisateurs d'un même réseau local,
– évoluer peu dans le temps.

L'introduction de ces contraintes nous permet d'obtenir des systèmes anonymisant des communications hautement interactives, dont le paradigme sera tout au long de ce mémoire la voix sur IP.

Le premier chapitre présente dans un premier temps les notions de base et les primitives classiques servant à l'anonymisation des communications, ainsi que les problèmes propres à l'utilisation de ces primitives. Dans un deuxième temps, nous analysons les problèmes liés aux performances des systèmes d'anonymisation des communications, et nous essayons de mettre en évidence l'origine de ces problèmes. Enfin, nous décrivons le cadre de travail dans lequel nous nous plaçons

pour le restant du mémoire.

Le deuxième chapitre est dédié à la présentation de deux serveurs basés sur les primitives classiques pour l'anonymisation des communications. Dans un premier temps nous présentons un type de serveur, que nous appelons *EBBS*, basé sur l'émission de bourrage chiffré et la diffusion avec adressage implicite, puis nous étudions ses performances. Ensuite, nous proposons un deuxième type de serveur, que nous appelons *serveur DC-net*, basé sur l'envoi superposé et la diffusion avec adressage implicite. Divers protocoles sont présentés pour rendre un tel serveur viable, avant de présenter et d'analyser les performances d'une implémentation que nous avons réalisée.

Dans le troisième chapitre, nous introduisons les protocoles de récupération d'informations privée (ou protocoles PIR) dans le but de les utiliser comme une nouvelle primitive permettant d'anonymiser les communications. En premier lieu, nous présentons les notions de base relatives à ces protocoles. Ensuite, nous analysons, en respectant l'ordre chronologique, chacun des protocoles qui de nos jours sont disponibles en mettant en relief les apports de chacun de ces schémas par rapport à ses prédécesseurs. Enfin, nous comparons les performances de chacun d'entre eux. L'annexe A contient une série de définitions et les preuves de sécurité que nous avons réalisées sur les protocoles PIR.

Le quatrième chapitre détaille les nouveaux systèmes qui peuvent être réalisés pour anonymiser des communications en utilisant les protocoles PIR comme primitive. Dans un premier lieu, nous modifions un serveur EBBS petit à petit jusqu'à obtenir un système efficace basé sur l'émission de bourrage chiffré et sur les protocoles PIR, que nous appelons pMIX. Suite à une étude de performances du pMIX, nous présentons différentes variantes de celui-ci résultant de la combinaison des primitives classiques avec les protocoles PIR, ainsi qu'une étude de performances pour chacune de ces variantes. Enfin, nous présentons une taxonomie des systèmes permettant d'anonymiser les communications dans le but d'illustrer en quoi l'introduction des protocoles PIR comme primitive, nous permet d'élargir considérablement les options disponibles pour implémenter de tels systèmes.

Ce mémoire se conclut par un résumé des résultats acquis au cours de cette étude, et une ouverture vers d'autres tests complémentaires, des implémentations, et des axes de recherche dans le cadre de la récupération d'informations privée.

Chapitre 1

Problématique et état de l'art

1.1 L'analyse du trafic

Il est en général facile d'obtenir le contenu des paquets formant une communication et circulant dans un réseau informatique :

– par écoute passive, par exemple en se branchant physiquement sur un câble du réseau ou en analysant le rayonnement électromagnétique des équipements du réseau ;
– en compromettant les équipements du réseau (routeurs, switchs, proxies) pour rediriger les paquets ou une copie des paquets vers la machine de l'attaquant[1].

Pour éviter qu'un attaquant ne puisse lire le contenu de paquets qui ne lui sont pas destinés on peut chiffrer[2] ce contenu. Mais même si le contenu du paquet est chiffré, un attaquant peut encore obtenir certaines informations comme l'existence d'un trafic, ainsi que l'adresse de l'émetteur ou du récepteur d'un paquet simplement en analysant son en-tête. Ces informations son dites de *traçage*, et

[1] Il est possible, par exemple, de tromper un switch en utilisant sa fonction d'apprentissage.

[2] Dans ce mémoire, nous faisons systématiquement l'hypothèse que le chiffrement est parfait, c'est-à-dire que sans connaître la clé de déchiffrement les attaquants ne peuvent extraire aucune information d'un message chiffré, si ce n'est son existence. En particulier, les attaquants ne peuvent pas apprendre qui a créé un message ou à qui il est destiné simplement analysant les message chiffré. De plus, quand nous utilisons des systèmes de chiffrement à clé publique nous présupposons l'existence d'une infrastructure associée permettant d'obtenir les clés publiques de tous les utilisateurs.

l'obtention et analyse de celles-ci est appelée *analyse du trafic*.

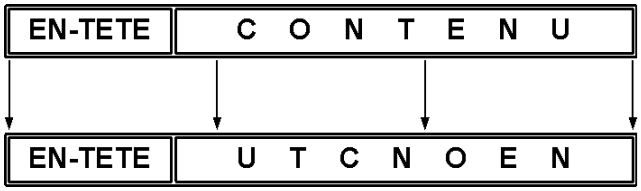


FIG. 1.1: Exemple du chiffrement du contenu d'un paquet

Ce problème ne peut pas être résolu par le chiffrement de l'ensemble du paquet, y compris l'en-tête. En effet, l'en-tête des paquets doit être conservé en clair, puisque les informations qu'il contient sont nécessaires au routage. La figure 1.2 illustre ce problème. Pour empêcher l'analyse du trafic, il faut donc mettre en oeuvre des solutions plus complexes, qui sont présentées dans la suite de ce chapitre.

FIG. 1.2: Chiffrement des en-têtes

1.2 Les communications anonymes

Un utilisateur peut souhaiter envoyer un message isolé à un destinataire donné. Il peut aussi vouloir envoyer plusieurs messages à la suite au destinataire, ou espérer recevoir des réponses, à ses messages. Les réponses peuvent arriver longtemps

après l'envoi de ses messages ou de façon presque immédiate, et dans certains cas les réponses peuvent même être entrelacées avec les envois. La difficulté pour rester anonyme dans chacun de ces cas est croissante, et malheureusement en informatique le cas le plus courant est le dernier. Dans le cas d'envois de plusieurs messages avec éventuellement des réponses entrelacées avec ceux-ci dans le temps, on parlera de *communications bidirectionnelles* ou simplement de *communications*[3]. Ainsi, ce terme couvre les communications de type voix ou vidéo-conférence sur IP, mais aussi la navigation Web, ou les connexions de type TCP.

Dans ce qui suit, nous poserons quelques définitions générales, que nous illustrerons, pour simplifier, par des exemples utilisant des envois de messages indépendants. Lorsque l'enchaînement de messages ou l'existence de réponses pose un problème spécifique, cela sera discuté. En particulier, la section 1.4 est dédiée à certains des points les plus importants à prendre en compte dans ces situations.

Lors d'une communication nous appelons *interpellant* l'utilisateur ayant initié celle-ci, et *interpellé* son interlocuteur.

Une communication qui suit un protocole protégeant les utilisateurs contre l'analyse du trafic est dite *communication anonyme*. Nous présentons dans la section 1.2.1 les notions de base nécessaires à la description de tels protocoles ainsi que les propriétés d'anonymat qu'on peut espérer obtenir en les utilisant.

1.2.1 Notions de base

1.2.1.1 Définitions

Nos définitions sont basées sur la terminologie proposée par Pfitzmann et Köhntopp [PK00].

Deux éléments sont dits *non-associables* pour un attaquant *A*, si *A* n'a pas plus d'information sur les liens entre ces éléments après ses attaques qu'il n'en avait avant. La plupart des notions d'anonymat se définissent simplement comme l'impossibilité d'associer des messages ou des actions aux utilisateurs.

L'anonymat est l'état d'une personne ou d'une chose dont on ne connaît pas le

[3] Dans la suite de ce mémoire, lorsqu'on voudra faire référence à des communications non-interactives ce sera dit explicitement.

nom. En informatique, on définit plutôt l'*anonymat* comme la propriété qu'un utilisateur ne puisse pas être identifié parmi un ensemble de sujets qu'on appelle le *groupe anonyme*[4]. Il est important de remarquer que cette définition de l'anonymat n'est pas absolue mais relative à un groupe et en particulier à sa taille. Ainsi il est très différent d'être anonyme au sein d'un groupe de deux, de cent ou de dix-mille utilisateurs.

Enfin, quand un ensemble de serveurs permettent de garantir des communications anonymes à leurs utilisateurs, nous appelons cet ensemble le *fournisseur d'anonymat de communication* (FAC).

1.2.1.2 Remarques

Si un utilisateur se connecte et se déconnecte d'un FAC en début et en fin de chaque communication anonyme, alors il est facile d'observer *quand* et *combien* de fois par jour il réalise des communications pour lesquelles il veut se protéger des curieux. Pour éviter cela, il vaut mieux se maintenir connecté au FAC, même en absence de communication.

Dans la plupart des systèmes, plus il y a de clients, plus les groupes anonymes sont grands et donc meilleur est l'anonymat fournit par le FAC. Ainsi, un client du FAC, peut être quelqu'un qui est en train de communiquer anonymement, ou bien quelqu'un qui reste connecté longtemps au FAC pour diminuer la visibilité de ses communications, voire tout simplement quelqu'un qui désire collaborer avec les autres utilisateurs pour agrandir la taille du groupe anonyme, et ainsi contribuer à fournir un meilleur service d'anonymat.

Notons que les clients se fournissent mutuellement un service d'anonymat par le simple fait de former entre eux des groupes anonymes. Ainsi, la séparation FAC / clients du FAC, n'est pas toujours marquée et peut être même inexistante (cf. 1.5.3 et [RR98, LS02])

[4] Quand sous certaines conditions exceptionnelles (litige commercial, décision de justice, etc.) on peut rétroactivement associer des actes à un utilisateur, on parle plutôt de pseudonymat.

Exemple 1.1 Envoi de courriels

Tous les matins, Alice, Bob et Charlie vont au cybercafé. Ils utilisent tous le même ordinateur à tour de rôle. Dans l'ordre, Alice, Bob et Charlie tapent chacun un courriel pendant exactement cinq minutes. Si quelqu'un finit son courriel en moins de cinq minutes il tape du charabia pendant le temps restant. Enfin, Charlie envoie dans un ordre aléatoire les trois courriels à partir de l'adresse `alicebobetcharlie@anonymous.net`.

Quelqu'un qui observerait les utilisateurs dans cet exemple, verrait trois personnes qui font tous les matins toutes les trois la même chose. Écouter le câble qui va de l'ordinateur vers l'Internet permettrait de voir ce qui a été écrit, mais pas qui a écrit quoi. Ces utilisateurs ont un anonymat d'émission et leur groupe anonyme est de taille trois vis-à-vis d'un attaquant extérieur.

Cet exemple reflète une notion extrêmement importante dans les communications anonymes :

> **pour rester anonyme au sein d'un groupe, il faut que, vu de l'extérieur, tous les éléments du groupe agissent de façon similaire.**

Ceci n'est pas une condition suffisante pour être anonyme mais c'est la base de tous les protocoles qui de nos jours fournissent un anonymat de communication.

Dans ce protocole il n'y a pas de FAC autre que les utilisateurs eux-mêmes, qui se fournissent de l'anonymat les uns aux autres. En revanche, les membres du groupe anonyme peuvent trahir leurs collègues facilement. Dans les protocoles réels, comme nous le verrons plus loin (cf. 1.5.3), on peut se prémunir contre ce risque, de telle sorte que pour identifier quel utilisateur a envoyé un message, il faut que tous les autres utilisateurs – sans exception – le trahissent.

On peut remarquer qu'une hypothèse nécessaire à ce protocole, ainsi qu'à tout protocole destiné à fournir un service d'anonymat de communication, est que personne "ne regarde par dessus l'épaule" de la personne essayant de communiquer anonymement. Dans le monde informatique, ceci se traduit en général par l'absence de logiciels espions ou d'intrus au sein de l'ordinateur utilisé. Cette hypothèse est faite implicitement tout au long de ce mémoire.

1.2.2 Propriétés d'anonymat

Pour un groupe donné d'utilisateurs G, on dit qu'il y a *anonymat d'émission* au sein de G – ou que G est un *groupe anonyme d'émission* – vis-à-vis d'un attaquant A, si pour A il y a non-associabilité entre les membres de G et les messages émis par ceux-ci. Lorsque le groupe anonyme d'émission contient des utilisateurs qui ne sont pas en train d'émettre on dit qu'il y a *non-observabilité d'émission*, et que G est un *groupe d'émission non-observable*.

On définit de façon analogue, *anonymat de réception*, *groupe anonyme de réception*, *non-observabilité de réception*, et *groupe de réception non-observable*.

Enfin, on dit qu'il y a *anonymat relationnel* entre deux groupes d'utilisateurs G_1 et G_2 vis-à-vis d'un attaquant A, quand pour A il y a non-associabilité entre les membres de G_1 et ceux de G_2.

La distinction qui est faite entre G_1 et G_2 est importante. Les deux groupes sont souvent différents (mais rarement disjoints). Par exemple dans beaucoup de systèmes G_1 est l'ensemble des clients du FAC alors que G_2 est la réunion des clients du FAC et des utilisateurs interpellés par les clients du FAC.

Les propriétés d'anonymat ou de non-observabilité d'émission, et de réception, sont étroitement liées entre elles et à l'anonymat relationnel. Dans un groupe G d'utilisateurs non-observables en émission et en réception il y a forcément un anonymat relationnel. Cependant, les relations entre ces propriétés dépendent du système étudié et de l'attaquant considéré. Le lecteur est invité à lire [PK00] pour une présentation plus précise des concepts qui sous-tendent les définitions des propriétés d'anonymat et leurs relations.

À chacune de ces trois propriétés, correspond une multitude d'exemples concrets où il est important qu'une de celles-ci soit respectée. Par exemple, l'anonymat d'émission peut permettre à un individu d'échapper à la censure d'un gouvernement, ou encore à un employé de notifier sans risque la violation de lois ou codes déontologiques par des membres de son entreprise. L'anonymat de réception peut permettre d'obtenir des conseils sur un thème sensible (Alcooliques Anonymes, maladies graves,...) sans craindre de se faire identifier comme destinataire de ces conseils. Enfin une liaison intime ou une collaboration confidentielle peuvent être occultés par l'utilisation d'un système fournissant de l'anonymat relationnel.

1.2.3 Modèles d'attaquant

Il existe deux modèles d'attaquant classiques dans le contexte de l'anonymat de communication. L'*observateur local* peut voir les paquets entrants et sortants d'un ou de plusieurs ordinateurs donnés. L'*observateur global*, quant à lui, peut voir les paquets circulant n'importe où dans le réseau. Ces attaquants peuvent être *passifs* et se contenter d'observer les paquets qui circulent, et quand ils sont émis ou reçus, ou bien il peuvent être *actifs*, c'est-à-dire qu'ils peuvent aussi insérer, effacer et modifier les paquets[5].

Dans certains cas, on tient compte d'attaquants plus précis, qu'on ne définit pas par rapport à leur capacités mais par rapport à leur fonction. Ainsi, l'interlocuteur interpellé et les responsables d'un FAC doivent parfois être considérés comme attaquants possibles. Par exemple, un utilisateur peut vouloir préserver son anonymat vis-à-vis d'un serveur pour éviter qu'on lui associe un profil, ou vouloir utiliser un FAC sans faire confiance aux administrateurs du système.

Un modèle d'attaquant plus récent est celui de l'interpellant. En effet, dans certains cas, on peut vouloir fournir un service anonymement, c'est-à-dire en préservant l'anonymat du site fournissant le service même vis-à-vis de ses clients. On dit des protocoles fournissant cette possibilité qu'ils autorisent la création de *points de rendez-vous anonymes*.

Enfin, on peut définir un attaquant par rapport à ses motivations. Parmi tous les cas possibles, on s'intéressera plus particulièrement à un attaquant à la fois commun et puissant : il s'agit de l'attaquant qui a un soupçon et qui souhaite le confirmer, c'est-à-dire l'*attaquant suspicieux*.

[5] Tout comme les cryptosystèmes, les primitives cryptographiques tels que les générateurs pseudo-aléatoires, ou les fonctions de hachage sont considérées invulnérables quel que soit l'attaquant.

1.3 Les relais

1.3.1 Utilisation d'un relais

Si dans la vie courante, nous voulons que quelque chose soit fait, et nous ne voulons pas être vus en train de le faire, il est possible de demander à quelqu'un d'agir pour nous. En particulier il est possible de demander à quelqu'un de servir d'intermédiaire, ou *relais*, pour l'envoi ou la réception de messages.

Exemple 1.2 Relais de lettres.

Alice propose de servir de relais auprès d'autres utilisateurs. Bob veut envoyer une lettre à une amie, Laure. Il écrit la lettre, l'introduit dans une enveloppe où il a écrit l'adresse de Laure, puis introduit cette première enveloppe dans une autre enveloppe qu'il scelle et envoie à Alice (voir figure 1.3.1).

Quand Alice reçoit une lettre elle déchire l'enveloppe extérieure et envoie l'enveloppe contenue à Laure par la poste. Cette dernière enveloppe portera alors le cachet de la poste d'Alice, et non pas celui de la poste de Bob.

Un attaquant qui examinerait le courrier posté par Bob ne verrait que des lettres destinées à Alice et ne saurait pas qui est le destinataire final. De même un attaquant qui examinerait le courrier posté par Alice ne saurait pas quel est l'émetteur initial. En revanche, un attaquant qui observerait le courrier reçu et émis par Alice verrait que peu de temps après l'arrivée d'une lettre de Bob, Alice a posté une lettre destinée à Laure et pourrait en déduire que probablement Bob a envoyé une lettre a Laure en utilisant Alice comme intermédiaire.

En informatique, on n'utilisera pas un sceau comme dans l'exemple introduit ci-dessus pour empêcher à un attaquant d'ouvrir l'enveloppe extérieure, mais une opération cryptographique. Les opérations exactes et l'ordre de celles-ci dépendent du système. Ainsi on peut avoir :
– un chiffrement par l'émetteur et un déchiffrement par le relais ;
– un chiffrement par le relais et un déchiffrement par le destinataire ;
– un déchiffrement puis un chiffrement avec une autre clé par le relais ;
– etc.

Dans certains cas on utilisera des cryptosystèmes à clé publique et dans d'autres des cryptosystèmes à clé secrète. Dans les sections 1.3.5 et 1.3.4 le lecteur trouvera des exemples pour quelques uns de ces cas.

En pratique, utiliser un seul relais peut présenter plusieurs inconvénients :

– si le relais décide de nous trahir ou de garder un journal de tout ce qu'il fait pour d'autres personnes nous pouvons nous retrouver à découvert sans même en être notifié ;
– il est assez simple d'observer les actions du relais et de relier ce qu'il reçoit et ce qu'il envoie, ce qui nuit à l'anonymat des utilisateurs.

C'est pour ces raisons que le plus souvent on utilise plusieurs de ces relais à la suite. Ainsi par exemple si un utilisateur se sert de deux relais, le premier sait uniquement qu'il communique avec le second relais, alors que le second relais sait uniquement que le premier relais communique avec le destinataire. Ce n'est que si les deux relais s'entendent secrètement pour trahir leurs utilisateurs et mettent leurs informations en commun qu'ils pourront associer l'émetteur au destinataire. Il suffit donc de choisir des relais qui ont des intérêts divergents, ou qui n'ont aucune relation entre eux, pour que cette collaboration (qui serait alors une *collusion*) contre l'utilisateur soit peu probable.

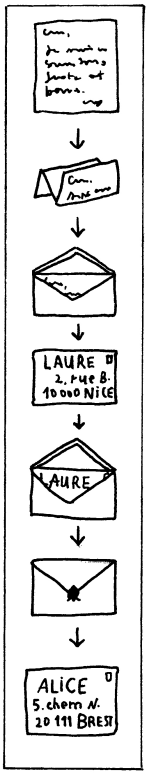


FIG. 1.3: Encapsulation

Sans collusion avec les relais, un attaquant doit observer à la fois ce que font le premier et le deuxième relais, ce qui n'est pas facile par exemple s'ils ont été choisis éloignés géographiquement ou dans des zones administratives indépendantes. Bien sûr, l'utilisation de plusieurs relais limite la bande passante à celle du lien le plus faible et augmente la latence des communications. C'est pour cela que le nombre de relais dépend du degré de sécurité voulu, mais aussi des performances requises.

1.3.2 Organisation des relais

1.3.2.1 Le relais isolé

Dans certains cas [ANZ, Hel95], un unique relais (voir figure 1.4) propose ses services pour les clients qui veulent communiquer anonymement. Si l'on veut juste établir une connexion unique et qu'on souhaite se protéger d'un attaquant local ou du serveur sur lequel on se connecte, un tel relais offre un anonymat suffisant et peut se révéler utile. Cependant, un effort de recherche important est fait dans l'étude des systèmes utilisant plusieurs relais. En effet ces systèmes peuvent fournir un anonymat de communication de telle sorte que les informations de traçage ne soient pas accessibles à un unique relais, qui pourrait les journaliser et les divulguer sans le consentement de l'utilisateur.

On peut remarquer que dans le cas où une seule entreprise contrôle un ensemble de relais, l'utilisation de plusieurs de ses relais successivement dans une même communication rend les attaques extérieures plus difficiles, mais présente un risque important de collusion entre les relais, ce pourquoi nous assimilons cette approche à celle d'un relais unique.

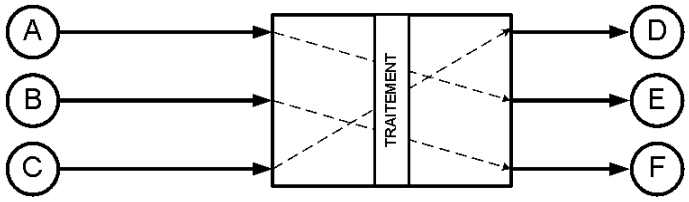


FIG. 1.4: Relais isolé.

1.3.2.2 Les cascades

Les relais peuvent s'organiser en *cascades* (voir figure 1.5), ou chaînes de relais fixées au préalable. Ainsi certains systèmes proposent aux utilisateurs d'utiliser de telles cascades de relais (on peut choisir parmi plusieurs cascades mais pour une cascade les relais sont fixes), chaque relais étant géré par une entité différente (en général une entreprise ou une organisation pour la protection de la vie privée). Nous décrirons plus longuement l'un de ces systèmes dans la section 1.3.5.

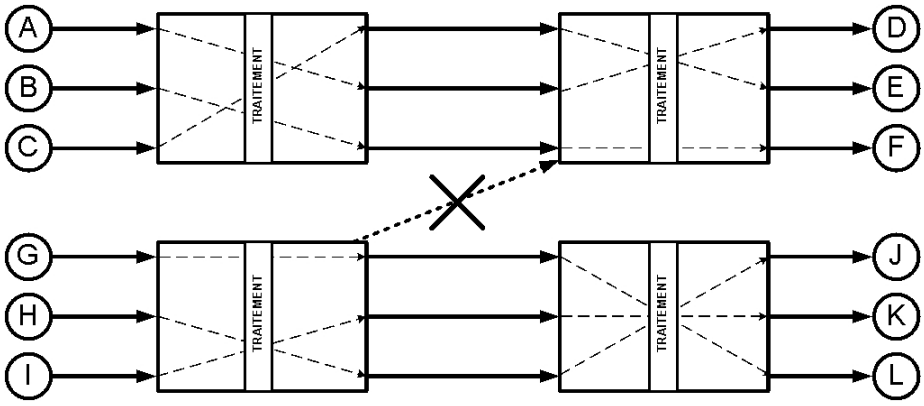


FIG. 1.5: Relais en cascades.

1.3.2.3 Les réseaux de relais

On dit que des relais forment un réseau (voir figure 1.6) quand les chaînes de relais ne sont pas fixées au préalable comme dans le cas des cascades. Si le choix d'un relais en amont limite le choix des relais suivants on dit qu'il s'agit d'un réseau de relais à routage restreint [RR98, FM02]. Si le choix d'un relais ne limite en rien le choix des relais ultérieurs on dit que le réseau est à routage libre [DMS04, GRS99].

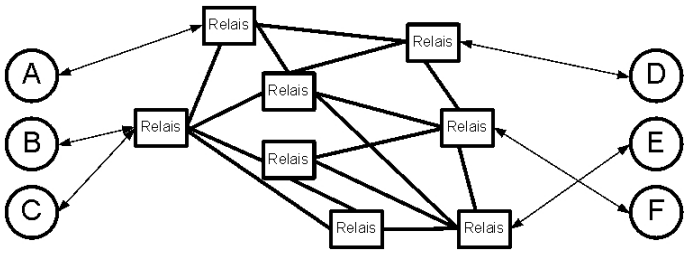


FIG. 1.6: Réseau de relais.

L'organisation des relais est une caractéristique importante d'un système visant

à protéger l'anonymat de communication. Cependant, le relais étant l'élément de base du protocole, c'est son comportement qui va le plus fortement déterminer le type de protocole utilisé. Nous présentons dans ce qui suit deux exemples de relais qui sont utilisés de nos jours dans des systèmes réels fournissant un anonymat de communication.

1.3.3 Les MIX

Les premiers relais utilisés pour l'anonymat de communication furent décrits par David Chaum en 1981 [Cha81]. Ces relais, qu'il appela *MIX*, opèrent de la façon suivante :

– ils n'acceptent que des messages de taille fixe ;
– ils déchiffrent les messages entrants avec leur clé privée ;
– ils attendent d'avoir reçu un certain nombre de messages ;
– puis réordonnent les messages avant de les réémettre.

L'objectif est de fournir un anonymat relationnel entre le groupe des utilisateurs émetteurs d'un message, et le groupe des utilisateurs récepteurs d'un message vis-à-vis d'un attaquant qui observerait tous les messages arrivant et sortant du MIX.

Si nous reprenons l'exemple du service d'envoi de lettres fourni par Alice dans la section précédente (exemple 1.2), nous disons qu'Alice se comporte comme un MIX si elle n'accepte de rendre son service que pour des lettres comportant un nombre fixe de pages, attend d'avoir un certain nombre de lettres avant de les envoyer, et elle les mélange aléatoirement avant d'aller les poster. Un attaquant n'a plus aucun moyen de faire le lien entre les messages entrants et sortants, et du coup ne peut faire le lien entre un émetteur et un récepteur (voir figure 1.7).

1.3.3.1 Fonctionnement d'un MIX

Quand un utilisateur A veut envoyer un message M à un utilisateur E, il ajoute l'adresse de E au message et chiffre le tout avec K_{MIX} la clé publique du MIX. On note ce message chiffré $K_{MIX}(R_A, M, E)$, R_A étant un *nonce*[6].

[6] Valeur aléatoire utilisée une seule fois.

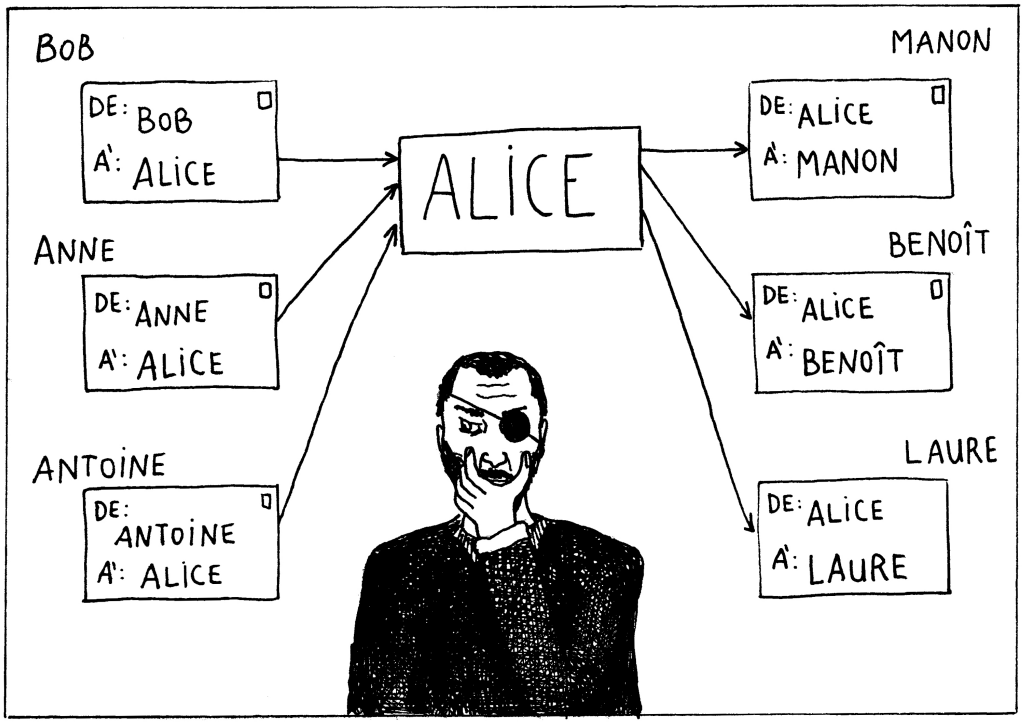


FIG. 1.7: Service de type MIX.

Quand le MIX reçoit le message chiffré, il le déchiffre avec sa clé privée, élimine le nonce R_A, et met en attente le résultat M, E. Quand le MIX a mis en attente un nombre donné de messages[7] il les envoie dans un ordre aléatoire aux adresses de destination. La figure 1.8 illustre ces opérations.

S'il y a peu de trafic, les délais introduits par un MIX de ce type peuvent être excessifs. Pour diminuer les délais de transit, de faux messages peuvent être envoyés au MIX, voire insérés par le MIX lui-même pour assurer que les tampons se remplissent assez fréquemment.

En supposant que tous les utilisateurs envoient des messages de même taille, le MIX décrit ci-dessus fournit un anonymat relationnel vis-à-vis d'un attaquant

[7] Une autre façon de cacher un message parmi les messages reçus juste avant ou juste après lui consiste pour le MIX à attendre un temps aléatoire avant de réémettre chaque message [DS03].

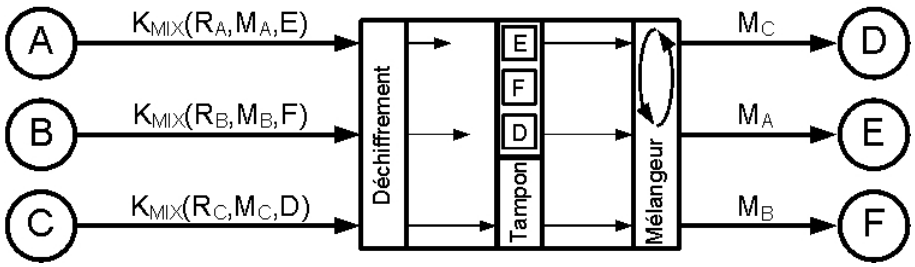


FIG. 1.8: MIX avec un tampon pour trois messages.

passif qui pourrait observer tous les liens (observateur global). Cependant des attaques actives sont toujours possibles, et nous décrivons dans les sections 1.3.3.3 et 1.3.3.4 deux des plus connues.

1.3.3.2 Utilisation de plusieurs MIX

Quand un utilisateur veut utiliser plusieurs MIX les uns après les autres il va appliquer plusieurs couches de chiffrement pour qu'elles soient traitées et retirées à raison d'une par MIX utilisé. Par exemple, un utilisateur A qui veut envoyer un message M à un utilisateur U va chiffrer une première fois le message pour le MIX le plus proche du destinataire, qu'on nomme $MIX2$ et obtenir $K_{MIX2}(R_A, M, U)$. Il va ensuite re-chiffrer ce message à l'intention d'un autre MIX, $MIX1$ précédant $MIX2$ dans le chemin de A à U, pour obtenir :

$$K_{MIX1}(R_{A'}, K_{MIX2}(R_A, M, U), MIX2)$$

Ces MIX peuvent être organisés en cascades (voir figure 1.5) ou en réseau (voir figure 1.6). La figure 1.9 illustre l'exemple donné ci-dessus pour deux MIX en cascade .

1.3.3.3 Attaque par rejeu

On peut remarquer que le MIX opère de façon déterministe, et donc que si un message entrant X produit un message sortant Y vers un utilisateur U, toute nouvelle réception du message X provoquera la réémission du même message sortant

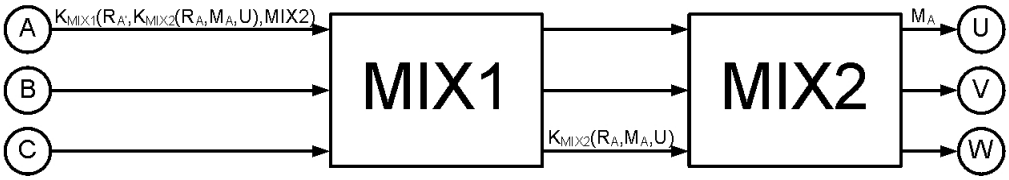


FIG. 1.9: Utilisation de deux MIX en cascade.

Y vers l'utilisateur *U*. Un attaquant peut donc tenter d'envoyer à un MIX un doublon d'un message entrant précédent (c'est-à-dire rejouer un message) pour repérer les doublons en sortie (voir figure 1.10). Pour cette raison les MIX surveillent l'arrivée de doublons et les éliminent le cas échéant.

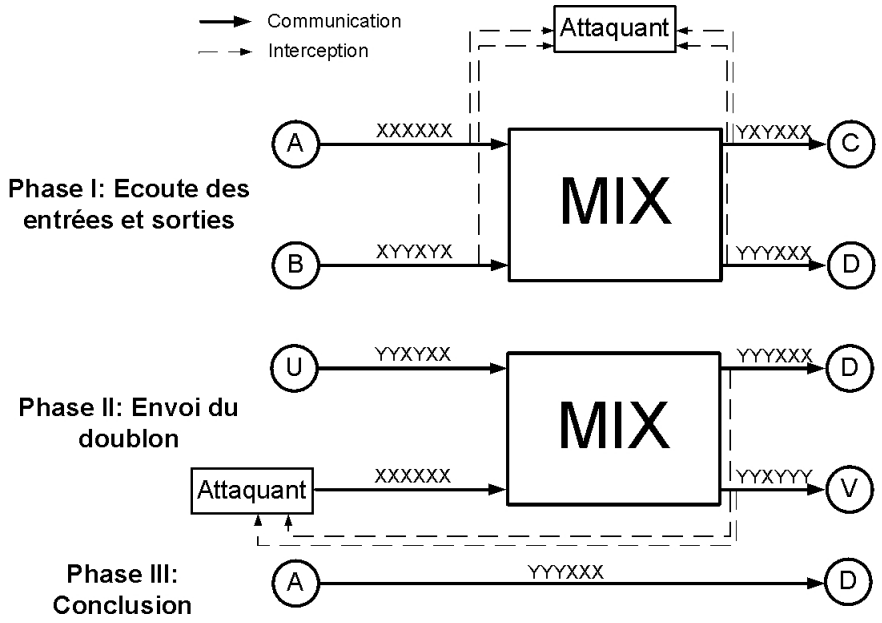


FIG. 1.10: Attaque par rejeu.

1.3.3.4 Attaque par inondation

Si un MIX attend d'avoir reçu n messages avant de les réémettre, un attaquant peut envoyer $n-1$ messages puis analyser la sortie où il peut repérer le message qui n'est pas parmi les siens. Bien sûr cette attaque n'a pas besoin d'être poussée jusqu'aux $n-1$ messages, il suffit qu'un attaquant envoie $\frac{\alpha-1}{\alpha} \times n$ messages pour réduire la taille des groupes anonymes d'un facteur α.

Pour ce type d'attaques comme pour d'autres, diverses parades ont été élaborées. Pour une présentation plus détaillée des MIX récents, le lecteur est invité à consulter l'étude présentée dans [DP04].

1.3.3.5 Utilisation du nonce

Deux raisons font que l'utilisation de le nonce est indispensable. La première est que si on n'utilise pas le nonce R_A, un attaquant pourrait reconstruire $K_{MIX}(M, E)$ à partir du message sortant M et de l'adresse du destinataire E (incluse dans l'entête du paquet sortant). Ceci lui permettrait de faire le lien entre le message entrant et le message sortant et donc entre l'émetteur et le destinataire.

La deuxième raison est que si R_A est différent de $R_{A'}$ on a :

$$K_{MIX}(R_A, M, E) \neq K_{MIX}(R_{A'}, M, E)$$

et donc les nonces permettent aux utilisateurs d'envoyer plusieurs fois le même message au même destinataire sans pour autant créer des doublons. Pour détecter les doublons, il suffit que le MIX conserve une table des nonces déjà reçus.

En pratique les nonce sont souvent formés de quelques bits aléatoires plus une estampille permettant de rejeter automatiquement les messages trop vieux et de ne maintenir qu'une liste de nonces récents.

1.3.4 Les *Onion Routers*

En 1997, des chercheurs du *NRL*[8] ont proposé l'utilisation d'un réseau de relais à routage libre (voir figure 1.6). Ces relais sont en fait des MIX que les auteurs

[8] *Naval Research Laboratory*, à Washington, États-Unis.

nomment *Onion Routers* (ORs). Ils maintiennent des connexions chiffrées permanentes entre eux au sein desquelles ils multiplexent les connexions des utilisateurs, et pour éviter l'analyse du trafic à l'intérieur de ces connexions ils utilisent du bourrage chiffré (cf. 1.5.2).

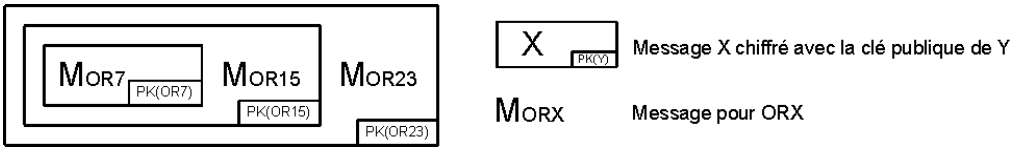


FIG. 1.11: Exemple d'oignon.

Lors de l'établissement d'une connexion le premier relais, qui est appelé l'*Onion Proxy* (OP), envoie un paquet le long d'une route qu'il choisit librement parmi les autres relais du réseau. La présence de multiples couches de chiffrement dans ce paquet sont à l'origine du terme *oignon* le désignant. L'oignon, illustré dans la figure 1.11, est créé par l'OP en ajoutant une couche de chiffrement[9] et un message pour chacun des ORs du chemin choisit par l'OP. Chacun des messages contient des informations pour l'établissement d'un canal permanent, et des clés de chiffrement symétrique. Une fois le canal formé le long de la route choisie, toutes les informations entre l'émetteur et le récepteur circulent à travers le canal ainsi créé avec une couche de chiffrement symétrique pour chaque OR traversé.

Pendant quelques années les auteurs ont proposé de façon expérimentale un petit réseau composé de quelques relais, tous exécutés au sein d'une même machine. Actuellement une deuxième génération dénommée Tor [DMS04] est proposée, avec un nombre de différences important.

Premièrement, les OR ne sont plus des MIX. Le réordonnancement des messages à été abandonné dans le but de diminuer le plus possible la latence introduite par le système, ainsi que le bourrage qui a été considéré comme trop coûteux [ADS03].

Deuxièmement, l'OP est un programme qui est exécuté au sein des ordinateurs des utilisateurs de Tor. Il consulte des serveurs (les *Serveurs de répertoire*) qui lui informent des ORs disponibles, relaye le trafic local des applications (qui dans le cas de la navigation Web est filtré auparavant par le proxy *Privoxy*, comme

[9] Le chiffrement est réalisé en utilisant les clés publiques des relais.

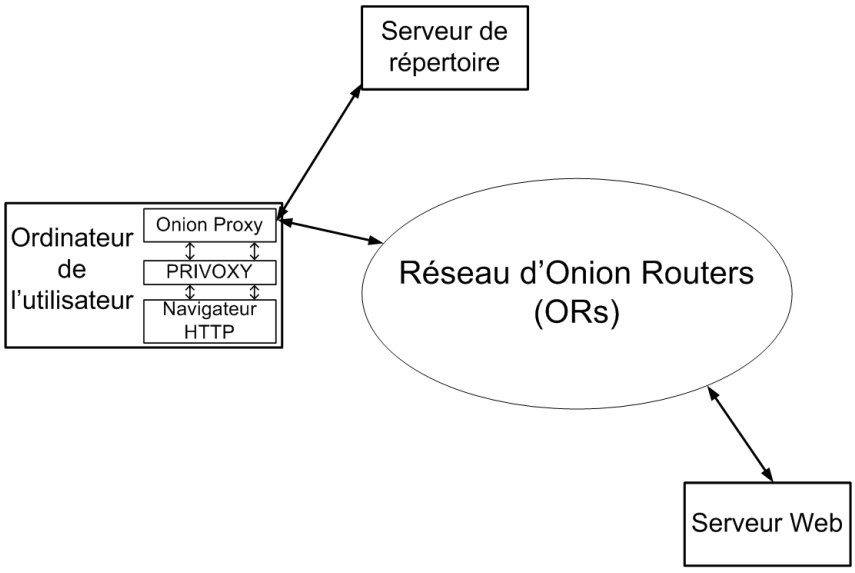


FIG. 1.12: Le fonctionnement des Onion Proxies (OP).

le montre la figure 1.12) vers le réseau des ORs et s'occupe de la création des canaux.

Enfin, la notion d'oignon n'existe plus : l'OP construit de façon incrémentale le canal qui va lui permettre de communiquer avec le récepteur. Pour cela les OR acceptent deux types d'opérations.

La première sert à créer un canal chiffré symétriquement et on la note <*Créer c*, $RSA_{OR}(DH(x_c))$>. $RSA_{OR}()$ représente un chiffrement RSA avec la clé publique de *OR*, et $DH(x_c)$ le premier des deux échanges nécessaires au protocole d'échange de clés de Diffie-Hellman [DH76]. A cette opération le serveur répond par un message qu'on note <*c Créé*, $DH(y_c)$>, $DH(y_c)$ représentant le deuxième des échanges nécessaire pour le protocole de Diffie-Hellman. Après la réponse de l'OR les messages dans le canal seront chiffrés avec AES à partir du secret commun obtenu par Diffie-Hellman, et on notera ceci par $c[m_{xxx}]$ (le message m_{xxx} est chiffré par AES dans le canal *c*).

La deuxième opération sert à prolonger un canal, et on la note <*Étendre c*, OR_k, $RSA_{OR_k}(DH(x_{c2}))$> avec OR_k étant l'OR vers lequel on désire d'étendre le canal *c*. Lorsqu'un OR reçoit un tel message il fait une demande de création de canal

vers OR_k en utilisant $RSA_{OR_k}(DH(x_{c2}))$ pour l'échange de clés par le protocole de Diffie-Hellman. Lorsqu'il obtient la réponse *<c2 Créé, $DH(y_{c2})$>*, il renvoie *<c Étendu, $DH(y_{c2})$>*. On remarquera que $DH(x_{c2})$ est chiffré avec la clé publique de l'OR final et non avec celle de l'OR intermédiaire qui est donc incapable d'obtenir le secret issu de l'échange par Diffie-Hellman. La figure 1.13 (extraite de [DMS04]) montre un exemple de création d'un canal et de son utilisation pour obtenir une page HTTP[10].

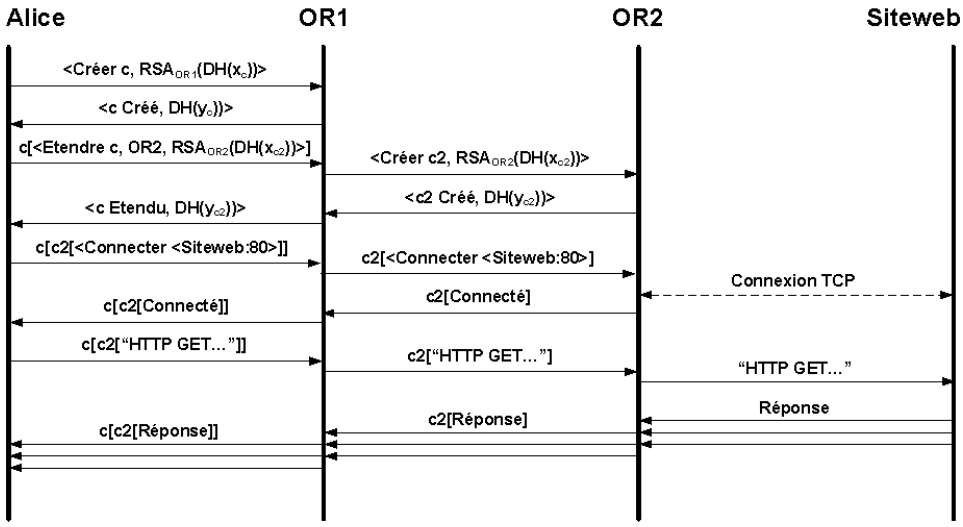


FIG. 1.13: Création et utilisation d'un canal.

Tor dispose actuellement de quelques centaines de relais distribués partout dans le monde. Il permet d'établir des connexions anonymes par SSH, par IRC ainsi que la navigation HTTP, entre autres. Un deuxième service fourni par Tor est la notion de points de rendez-vous qui permet d'avoir un service dont les utilisateurs ne connaissent pas la localisation. L'utilisation de Tor conduit à une latence lors de la navigation HTTP de quelques secondes. Les débits quand à eux, lors de transfert de fichiers sont très fluctuants (avec des pics à deux Mbits/s et des périodes de débit pratiquement nul pendant plusieurs secondes) mais la moyenne que nous

[10] Les communications indiquées dans la figure sont en plus chiffrées dans des tunnels TLS, autant entre les ORs qu'entre l'OP de l'utilisateur et le premier OR. Seules les communications entre le dernier OR et le serveur sont envoyées en clair.

avons mesuré à 580 Kbits/s est largement satisfaisante pour une connexion ADSL.

1.3.5 État de l'art

Depuis 1981 les MIX sont les relais d'anonymat qui ont donné lieu au plus grand nombre de publications. Ceux-ci ont beaucoup évolué et de façon générale on les distingue des autres types de relais par quatre particularités, qui découlent directement du fonctionnement des MIX initialement décrits par Chaum :

– le fait d'accepter des messages de taille fixe ;
– l'utilisation de fonctions de chiffrement et/ou déchiffrement sur les messages ;
– l'attente afin d'obtenir un nombre fixe ou variable de messages[DS03] ;
– le réordonnancement de ceux-ci.

Cet axe de recherche a donné lieu à un grand nombre d'articles, sur des attaques possibles [PP90, MK00, Ray00, BMS01, Dou02, Dan04, MD05], sur des variantes théoriques ou des implémentations des MIX proposés par Chaum [KEB98, DS03, DDM03, Jak99, GJ04b], et sur la mise en place de solutions complètes à grande échelle [PPW91, GT96, JMP+98, GRS99, FM02, RP04].

Le plus connu des FAC basé de nos jours sur des MIX et permettant d'avoir des communications bidirectionnelles à haute interactivité est JAP [DSDa]. JAP est un projet de recherche mené par les universités de Regensburg et de Dresde qui permet de naviguer dans le Web à travers plusieurs cascades de MIX. Souvent certaines de ces cascades offrent de très bonnes performances. Ainsi, lors de quelques tests rapides, nous avons obtenu des débits stables autour d'une centaine de Kbits/s lors de transferts de fichiers de taille importante, et un affichage des contenus avec une latence souvent inférieure à trois secondes lors de l'utilisation de moteurs de recherches . En revanche, lors de l'affichage de pages nécessitant de nombreuses requêtes HTTP, la latence est en général au delà des trente secondes, par rapport à quatre secondes environ lors d'une connexion non anonymisée[11]. Il est indéniable que JAP peut permettre à un utilisateur de naviguer dans le Web avec des performances acceptables, tant que les transferts sont de petite taille, que la navigation se fait sur des pages monolithiques, et qu'on accepte d'avoir des baisses d'interactivité sporadiques.

[11] Tests réalisés sur la page Web `http ://www.gt.tv`

D'autres types de relais ont été proposés dans la littérature, comme les *jondos* de Crowds [RR98] et Hordes [LS02]. Les jondos sont des relais qui décident par un choix aléatoire, soit d'envoyer les messages vers un autre jondo choisi au hasard ou de les envoyer directement au destinataire. Avec un tel système, même l'émetteur d'un message ne connaît pas le chemin que suivra un message ou le nombre de relais par lesquels il transitera. Les systèmes basés sur les jondos ne visent pas à fournir un anonymat fort. Le but de ces systèmes est que les utilisateurs qui en font partie puissent répudier un message qu'ils ont émis en se faisant passer pour un relais. Ces systèmes donnent une solution simple et élégante pour obtenir un certain anonymat vis-à-vis d'attaquants faibles. Ils ont été cependant abandonnés au niveau de la recherche et à notre connaissance ils ne sont pas utilisés en pratique de nos jours.

1.4 Attaques par corrélation

1.4.1 Corrélation au sein d'une communication

Les groupes anonymes évoluent dans le temps, et ceci peut avoir un impact important lorsque plusieurs messages sont associables entre eux. Par exemple deux messages appartenant à une même connexion TCP sortant d'un réseau local ont certainement été émis par le même utilisateur. Un attaquant saura que l'émetteur de ces deux messages est dans l'intersection du groupe anonyme d'émission du premier message et du groupe anonyme d'émission du deuxième message. Si ces groupes sont différents ceci résultera dans une diminution de l'anonymat de l'émetteur. Dans le cas des communications (même unidirectionnelles), des mesures précises peuvent mener à une perte totale de l'anonymat comme le montre la figure 1.14. Dans cet exemple cinq utilisateurs *A, B, C, D,* et *E* se servent d'un MIX pour obtenir des pages Web. Une simple observation ne suffit pas à savoir quel utilisateur parmi *A, B,* et *C* à demandé quelle page lors du premier tour. Lors du tour suivant, deux utilisateurs *B* et *C* n'envoient pas de message en laissant leur place aux utilisateurs *D* et *E*. En corrélant les informations sur les messages de chacun de ces deux tours (ici le destinataire des messages), un attaquant va pouvoir réaliser des intersections entre les groupes anonymes d'émission. Étant donné le grand nombre de sites Web existant et le fait qu'en général un utilisateur réalise plusieurs requêtes au même site, l'attaquant va pouvoir déduire que *A* est

probablement à l'origine des requêtes envoyées au site du LAAS.

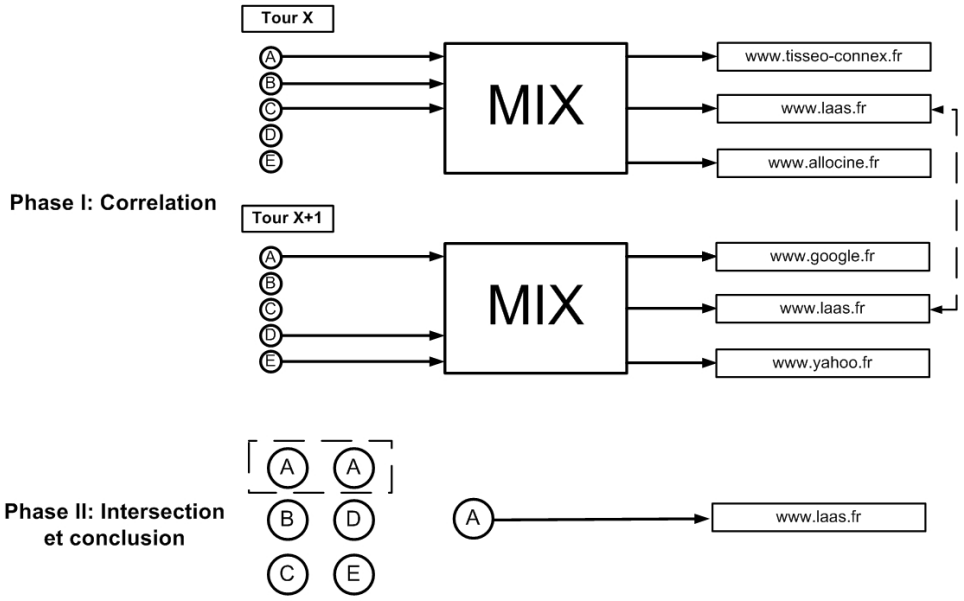


FIG. 1.14: Attaques par intersection.

Notons que dans cet exemple, deux choses rendent possible l'attaque. La première est que l'attaquant peut associer deux requêtes envoyées au LAAS comme des requêtes appartenant à une même communication. Dans de nombreux cas l'interlocuteur interpellé utilise des protocoles non-anonymisés (par exemple lors de la navigation HTTP), et l'ensemble des messages d'une communication peuvent être associés par n'importe quel attaquant. On supposera que c'est le cas dans le reste de ce mémoire, d'autant plus qu'il est possible d'associer des messages en se basant sur d'autres paramètres comme leur taille, le destinataire, la date d'émission, etc. La deuxième chose rendant possible cette attaque est que l'attaquant est capable de savoir quand un utilisateur émet.

Dans un système possédant des groupes anonymes d'émission il y a deux types d'utilisateurs : les émetteurs et les utilisateurs n'émettant pas. Dans un système possédant des groupes d'émission non-observable un attaquant ne peut pas distinguer entre les émetteurs et les utilisateurs n'émettant pas. Ainsi, les intersections

devront être réalisées sur l'ensemble des utilisateurs connectés (ici $\{A, B, C, D, E\}$ pour les deux tours), groupe qui est en général beaucoup moins changeant que les groupes anonymes d'émission (ici $\{A, B, C\}$ le premier tour et $\{A, D, E\}$ le deuxième), voire pas du tout.

Si un attaquant sait que certains messages sont plus probablement liés aux clients qui sont devenus membres du groupe non-observable récemment, par exemple parce que les clients ne se connectent au FAC que peu de temps avant de démarrer une communication, il pourra réaliser des attaques analogues à celle décrite ci-dessus. Même si ces attaques sont beaucoup moins efficaces que dans l'exemple, il est préférable, afin de réduire au maximum les informations qu'un attaquant peut tirer de telles attaques, que les utilisateurs essayent au maximum de décorréler la connexion au FAC et le début de leurs émissions (par exemple en se connectant tous les matins à la même heure, en attendant un délai variable après la connexion au FAC avant de démarrer une communication, etc.).

De la même façon, face à une telle attaque, un groupe anonyme de réception offrira peu ou pas de protection dans le cadre des communications, et les utilisateurs ont intérêt à ce que leur intégration dans un groupe de réception non-observable et les réceptions des premières communications soient aussi décorrélées que possible. La réception de communications par un utilisateur étant un acte en général involontaire ceci peut présenter des difficultés. Ce problème n'a pas de solution générale et devra être traité au cas par cas lors de l'implémentation d'une solution proposant de l'anonymat de communication.

1.4.2 Corrélation des émissions et réceptions

Dans le cadre des communications bidirectionnelles, les systèmes ne fournissant pas en même temps une non-observabilité d'émission et de réception sont vulnérables à la corrélation des données qu'un attaquant peut recueillir sur les émissions ou les réceptions. L'exemple suivant illustre ceci.

Exemple 1.3 Attaque par corrélation entre les émissions.

Soit un FAC tel que les utilisateurs émettant forment un groupe anonyme d'émission et qu'un groupe fixe d'utilisateurs (par exemple les utilisateurs d'un réseau local) forment un groupe de réception non-observable. À un instant donné, un utilisateur se met à émettre en utilisant le FAC. Un attaquant peut (sans rien apprendre sur le destinataire), enregistrer l'identité de l'émetteur, le moment où il a commencé à émettre et le moment où il finit d'émettre. Si l'attaquant observe qu'un deuxième utilisateur commence à émettre peu de temps après le premier, et finit en même temps ses émissions, il peut en déduire que probablement ces deux utilisateurs ont communiqué entre eux, brisant l'anonymat relationnel ainsi que la non-observabilité de réception.

De façon similaire, un FAC donnant lieu à un groupe non-observable en émission fixe et à un groupe anonyme en réception variable pourrait subir la même attaque par corrélation des réceptions de messages.

Soit A un attaquant. Soit G_1 le groupe d'émission non-observable d'un utilisateur U vis-à-vis de A, et G_2 son groupe de réception non-observable vis-à-vis de ce même attaquant. Si A réalise des corrélations entre les émissions et les réceptions des membres de G_1 et de G_2, les groupes d'émission non-observable et de réception non-observable de U se voient réduits à un même groupe qu'on appelle le *groupe non-observable de communication* ou (par abus de langage) *groupe anonyme de communication* qui est l'intersection de G_1 et G_2.

1.4.3 Les attaques par intersection

Les attaques présentés dans les sections 1.4.1 et 1.4.2 sont basées sur des corrélations à court terme. Quand les corrélations se font sur des analyses à long terme on parle d'attaques par intersection. Le cas le plus communément utilisé pour illustrer ces attaques est celui d'un attaquant qui observe comment le comportement d'un système change pendant les périodes de connexion d'un utilisateur donné. Deux modèles d'attaquants sont classiques dans ce cadre : l'observateur global, et l'attaquant suspicieux.

Associer les occurrences d'événements avec la présence d'un utilisateur est une

attaque puissante contre laquelle il est difficile de se protéger. Deux solutions sont possibles. La première est que le FAC soit distribué sur l'ensemble de l'Internet de façon à réduire au minimum le nombre d'attaquants qui pourraient observer le système d'un point de vue global. La deuxième est de disposer de FACs associés à des groupes d'utilisateurs stables de telle façon que les intersections seront peu efficaces. Ces deux alternatives et leurs avantages respectifs sont discutés dans la section 1.6.

1.5 Primitives pour la non-observabilité

Comme expliqué dans la section précédente, il est important que les utilisateurs puissent recevoir et émettre des messages de façon non-observable. Il n'existe dans la littérature qu'une seule primitive permettant de garantir la non-observabilité en réception : c'est la diffusion. En revanche, il existe deux primitives permettant la non-observabilité en émission : le bourrage chiffré et l'envoi superposé.

1.5.1 La diffusion

L'envoi de messages que tout le monde reçoit (ou peut recevoir) et qui n'est interprétable que par le destinataire est un moyen classique pour assurer la non-observabilité en réception (voir figure[12] 1.15).

En informatique, la diffusion permet d'envoyer un message à toutes les adresses d'un réseau ou d'un sous-réseau. Son utilisation est cependant contraignante puisque les liens de nombreux utilisateurs se voient encombrés, et même si on peut diffuser au delà d'un réseau local par l'utilisation de réseaux privés virtuels, il est impossible de diffuser des messages à très grande échelle (par exemple, sur l'ensemble de l'Internet). Nous proposons dans le deuxième chapitre de ce mémoire un système fournissant un service d'anonymat de communication, dérivé des protocoles à diffusion et avec un impact limité sur la charge réseau.

[12] Le deuxième exemple de cette figure touche aussi un point que nous ne traitons pas dans ce mémoire : comment inclure un message caché dans un message d'apparence anodine (la *stéganographie*).

FIG. 1.15: Utilisation de la diffusion

1.5.1.1 Les adresses implicites

Un protocole basé sur la diffusion ne peut être efficace que si les utilisateurs disposent d'un moyen pour identifier s'ils sont destinataires d'un message diffusé. Pour cela on utilise ce qu'on appelle l'*adressage implicite* [PW85] : on envoie avec le message diffusé une information sur l'identité du destinataire de façon à ce que seul celui-ci reconnaisse cette information. Ce type d'adressage est défini par opposition à l'*adressage explicite* où tout utilisateur peut associer les adresses aux destinataires (ce qui est le cas dans les protocoles non anonymisants).

Exemple 1.4 Protocole simple de diffusion (Version 1).

Les utilisateurs échangent deux à deux des pseudonymes. Quand l'utilisateur Alice veut envoyer un message à l'utilisateur Bob, elle indique au début du message le pseudonyme que Bob lui a donné, puis diffuse le message. Quand Bob reçoit le message, il reconnaît le pseudonyme et identifie ainsi que le message lui est destiné. Les autres utilisateurs ne reconnaissent pas le pseudonyme et ignorent le message.

Dans ce cas l'adresse implicite est le pseudonyme, tout le monde peut la voir et dans certains cas ceci peut être problématique. En effet, si Alice renvoie un mes-

sage quelques heures plus tard avec la même adresse implicite (le pseudonyme de Bob), tout le monde peut faire un lien entre le premier et le deuxième message. Cette associabilité entre les messages fournit des informations aux attaquants et peut être un problème. Quand on utilise des *adresses implicites visibles*, il faut les changer à chaque fois qu'on veut empêcher qu'on puisse associer de nouveaux messages aux messages passés, c'est-à-dire, dans le contexte où nous nous plaçons, à chaque nouvelle communication.

On remarque que dans ce protocole le contenu des messages est envoyé en clair, ce qui dans certains cas peut permettre à un attaquant d'identifier le destinataire d'un message à partir de son contenu. Pour éviter cela on peut chiffrer le contenu des messages. L'exemple suivant montre comment l'utilisation du chiffrement, permet de plus d'avoir des *adresses implicites invisibles*.

Exemple 1.5 Protocole simple de diffusion (Version 2).

Quand un utilisateur veut envoyer un message à un destinataire donné avec anonymat de réception, il le diffuse en chiffrant le contenu du message avec la clé publique du destinataire. Chaque utilisateur essaye de le déchiffrer avec sa clé privée, et si le résultat n'a pas de sens, il ignore le message.

Avec ce protocole, seul le destinataire va trouver un message ayant un sens lors du déchiffrement[13]. L'adresse implicite est définie par la capacité à obtenir un message ayant du sens ou respectant un certain format.

1.5.1.2 Utilisation d'un panneau d'affichage

Les utilisateurs peuvent aussi se servir d'un intermédiaire pour communiquer entre eux. Dans la vie courante on peut utiliser un panneau d'affichage où tout le monde écrit ses messages et le consulte régulièrement pour lire les messages des autres. C'est aussi le principe des petites annonces dans les journaux, qui sont ensuite diffusés chez les marchands de journaux. En informatique, on utilise le terme de système de panneau d'affichage ou *Bulletin Board System* (BBS) pour désigner un serveur qui maintient une base de données où les utilisateurs déposent

[13] Ce protocole peut fonctionner aussi avec un système de chiffrement à clé secrète, mais dans ce cas il faut que les utilisateurs partagent des clés communes deux à deux.

leurs messages, et qu'ils interrogent pour obtenir les messages des autres utilisateurs. Avant l'explosion du Web ce système était couramment utilisé pour diffuser des informations.

Il est facile d'obtenir une réception anonyme en utilisant un système d'affichage : quand un utilisateur veut envoyer un message, il l'inscrit dans la base. La réception des messages peut se faire soit par interrogation : les clients demandent régulièrement au serveur le contenu de toute la base de données, soit par diffusion périodique du contenu de la base vers tous les utilisateurs.

Ce protocole est un peu plus compliqué que les précédents et présente certains désavantages : la latence se voit augmentée, et il faut un serveur central. Cependant, certains avantages sont aussi à prendre en compte. Premièrement, il n'y a que le serveur qui réalise des diffusions, les utilisateurs eux envoient les messages uniquement en point-à-point au serveur. Ceci entraîne un diminution drastique des risques de collision, par exemple, dans un réseau Ethernet commuté. Deuxièmement, l'utilisation d'un serveur central simplifie l'implémentation de fonctions comme les réémissions en cas de perte de messages. Dans la pratique, ces avantages rendent en général préférable ce genre de solution.

Que les utilisateurs se servent d'un serveur comme panneau d'affichage ou qu'ils diffusent leurs messages dans un réseau, le principe de diffusion avec adressage implicite donne lieu a des groupes de réception non-observable composés de l'ensemble des membres du réseau (ou de l'ensemble des clients du serveur). Ceci est vrai quel que soit l'attaquant considéré.

1.5.2 L'utilisation de bourrage chiffré

Le *bourrage chiffré* consiste à envoyer des messages chiffrés dont le contenu en clair est nul (ou n'a pas de signification). Sous l'hypothèse que les attaquants ne peuvent pas distinguer les messages comportant des informations utiles de ceux qui sont du bourrage, on peut rendre les envois de messages ainsi que les augmentations brusques de volume inobservables pour un attaquant. JAP propose une option d'envoi de bourrage à ses utilisateurs.

Exemple 1.6 Utilisation de bourrage chiffré dans un MIX.

Soit un MIX qui possède *n* clients connectés de façon permanente et opérant par tours. À chaque tour, tous les clients envoient un message chiffré de même taille. S'ils n'ont rien à transmettre, ce message est du bourrage chiffré. À certains tours, il y a des messages qui sortent du MIX, à d'autres aucun.

L'utilisation de bourrage chiffré permet d'avoir des groupes d'émission non-observables avec des protocoles qui, en soi, ne fournissent pas cette propriété. Ainsi on peut améliorer les protocoles dans le cadre des communications bidirectionnelles et éviter les attaques par corrélation dans un grand nombre de situations.

Générer ce bourrage en émission est en général facile, il suffit que l'utilisateur envoie des messages qui en sortie du protocole d'anonymisation se révèlent inutiles de telle façon qu'ils sont éliminés. Quand l'utilisateur souhaite envoyer des informations utiles, il remplace des messages de bourrage par des messages contenant les informations. A cause du chiffrement, un attaquant ne peut pas différencier le bourrage des informations utiles et, tant que l'utilisateur n'envoie pas plus d'informations qu'il n'en envoyait lors du bourrage, l'émission sera non-observable.

Obtenir des groupes non-observables en réception avec du bourrage chiffré est autrement plus difficile. En effet deux possibilités s'ouvrent aux utilisateurs, générer ce bourrage eux mêmes et l'envoyer à travers le FAC ou recevoir du bourrage chiffré généré par autrui. Dans les deux cas nous ne connaissons pas de solution satisfaisante.

Générer ce bourrage soi-même implique un surcoût en émission pour l'utilisateur. Dans le cas ou l'utilisateur émet déjà du bourrage chiffré pour s'assurer des émissions non-observables les deux bourrages devront être assurés de façon séparée. En effet, utiliser le bourrage d'émission pour générer un bourrage en réception est en général une mauvaise idée : l'information utile en émission remplaçant le bourrage d'émission, il y aurait un chute dans le bourrage de réception si on confondait ces deux. De plus la réception de messages utiles se faisant de façon asynchrone pour l'utilisateur il ne pourra réduire le bourrage en réception qu'après une augmentation dans son flux d'entrée qui sera probablement détectable par un attaquant. Il semble difficile d'occulter ceci en générant soi-même le bourrage en réception.

Dans le cas ou le bourrage n'est pas généré par l'utilisateur le problème de l'augmentation du trafic lors de la réception de messages utiles se présente aussi sauf si l'émetteur du bourrage est celui qui envoie aussi l'information utile au récepteur. Cependant, dans ce cas la réception est observable pour l'émetteur du bourrage. Par exemple, le dernier relais d'une cascade peut envoyer du bourrage chiffré aux membres d'un groupe de réception non-observable. Cependant, dans ce cas la réception est observable pour l'émetteur du bourrage.

Nous ne voyons aucun moyen de générer ce bourrage en se basant uniquement sur les relais traditionnels. En effet, seul la génération du bourrage par le dernier relais semble viable, et dans le cadre des communications ceci implique qu'il n'y a pas d'anonymat face au dernier relais. En conclusion, face à un attaquant global, ou un attaquant suspicieux la confiance dans une chaîne de relais se voit réduite à la confiance dans le dernier relais de la chaîne. Étant donné que ceci est le plus souvent inacceptable, les systèmes basés sur des relais ont intérêt à rendre le modèle de l'attaquant global le plus invraisemblable possible. Les attaquants suspicieux, quant à eux, ont de grandes chances de confirmer leurs hypothèses.

1.5.3 L'envoi superposé

L'envoi superposé est un protocole selon lequel les utilisateurs se mettent d'accord pour tous émettre, même quand il n'ont rien à transmettre, en utilisant un format tel que la combinaison de toutes les émissions produit :

- un résultat nul, si aucun utilisateur n'a essayé de transmettre des informations,
- le message d'un utilisateur si un seul d'entre eux a essayé de transmettre,
- des messages brouillés entre eux si plusieurs utilisateurs ont essayé de transmettre en même temps.

En 1988, David Chaum a proposé le premier protocole d'envoi superposé tel que, quel que soit l'issue du protocole, on ne puisse rien apprendre sur les intentions de transmettre des utilisateurs [Cha88][14]. Pour décrire ce protocole, il a utilisé une parabole dans laquelle trois cryptographes se trouvent à une même table et veulent savoir si quelqu'un a payé, sans pour autant savoir lequel d'entre eux à réellement payé (c'est-à-dire que quelqu'un dise "J'ai payé" avec anonymat d'émission). Les

[14] Cet protocole était connu sous forme d'ébauche d'article depuis 1985, et une description incomplète avait été publiée dans [Cha85].

réseaux d'ordinateurs suivant ce protocole sont appelés les *réseaux du dîner des cryptographes* ou *DC-nets,* et le protocole est connu sous le nom de *protocole DC-net*.

1.5.3.1 Principe

Dans le protocole DC-net on utilise le fait que l'opération "ou-exclusif" (notée XOR) est involutive. En effet pour tous mots binaires a et b de même taille on a $a \oplus b \oplus a = b$, et si à cela on ajoute l'associativité et la commutativité on peut résumer ces propriétés par le fait que le XOR d'un nombre quelconque d'éléments élimine les éléments répétés un nombre pair de fois. Ainsi par exemple quel que soient a, b et c,

$$\begin{aligned} a \oplus b \oplus c \oplus b \oplus a \oplus c \oplus b \oplus a \oplus b &= a \oplus \not{b} \oplus c \oplus \not{b} \oplus a \oplus c \oplus b \oplus a \oplus b \\ &= \not{a} \oplus c \oplus \not{a} \oplus c \oplus b \oplus a \oplus b \\ &= \not{c} \oplus \not{c} \oplus b \oplus a \oplus b \\ &= \not{b} \oplus a \oplus \not{b} \\ a \oplus b \oplus c \oplus b \oplus a \oplus c \oplus b \oplus a \oplus b &= a \end{aligned}$$

Ceci permet de brouiller les messages envoyés par les utilisateurs et de choisir les brouillages de façon à ce qu'ils s'annulent les uns les autres lorsqu'on les combine. Nous décrivons ci-dessous un tour d'émission suivant le protocole DC-net. La compréhension exacte de ce mécanisme n'est pas nécessaire à la lecture de ce chapitre, mais est importante pour avoir une vue complète sur les différentes possibilités qui s'offrent à nous pour aboutir aux objectifs que nous nous sommes fixés.

Exemple 1.7 Envoi superposé de l bits sur un tour.

Soit un ensemble d'utilisateurs $U_1, \ldots, U_n$. Supposons que chaque couple d'utilisateurs (U_i, U_j) possède un secret commun $S_{i,j} = S_{j,i}$ de l bits.
Pour un utilisateur U_i on notera *Brouillage(i)* $= \bigoplus_{j=1, j\neq i}^{n} S_{i,j}$ et :
– *Message(i)* = 0...0 (l bits à 0) si U_i ne veut rien transmettre,
– *Message(i)* $= M_i$ (de l bits) si U_i veut transmettre le message M_i.

Déroulement du tour

1. Chaque utilisateur U_i émet :

$$\textit{Diffusion(i)} = \textit{Brouillage(i)} \oplus \textit{Message(i)}$$

2. Chaque utilisateur fait le XOR de toutes les diffusions et obtient le résultat du tour : *Résultat* $= \bigoplus_{i=1}^{n} \textit{Diffusion(i)}$

Les brouillages s'annulent mutuellement et les utilisateurs obtiennent :

$$\textit{Résultat} = \bigoplus_{i=1}^{n} \textit{Message(i)}$$

En effet, on a $\bigoplus_{i=1}^{n} \textit{Brouillage(i)} = \bigoplus_{i=i}^{n} \left(\bigoplus_{j=1, j\neq i}^{n} S_{i,j} \right)$, or chaque $S_{i,j}$ $(= S_{j,i})$ apparaît exactement deux fois, une fois introduit par U_i dans *Brouillage(i)*, et une autre par U_j dans *Brouillage(j)*. Les $S_{i,j}$ s'annulent donc tous deux à deux : $\bigoplus_{i=1}^{n} \textit{Brouillage(i)} = 0$. Comme

$$\textit{Résultat} = \left(\bigoplus_{i=1}^{n} \textit{Brouillage(i)} \right) \oplus \left(\bigoplus_{i=1}^{n} \textit{Message(i)} \right)$$

on obtient bien *Résultat* $= \bigoplus_{i=1}^{n} \textit{Message(i)}$.

Si un seul utilisateur U_i a essayé d'émettre un message, on a : *Résultat* $= M_i$. En revanche, si plusieurs utilisateurs $U_{i_1}, \ldots, U_{i_k}$ ont essayé de transmettre on a : *Résultat* $= M_{i_1} \oplus \ldots \oplus M_{i_k}$ c'est ce qu'on appelle une *collision*.

Une collision pourra être détectée par exemple si le message-résultat ne respecte pas un format donné. Dans le cas d'une collision, chaque utilisateur attend un nombre aléatoire de tours avant de ré-émettre, en espérant que la prochaine fois il

n'y ait pas de collision[15].

On peut remarquer que dans ce protocole l'utilisation de la diffusion fournit en plus un anonymat de réception si on le combine avec un système d'adresses implicites (cf. 1.5.1). Mais l'envoi superposé peut aussi être implémenté de façon à garantir uniquement un anonymat d'émission, comme le montre l'exemple suivant.

Exemple 1.8 Utilisation d'un serveur et d'adressage explicite.

Chaque utilisateur U_i calcule *Brouillage(i)* comme dans l'exemple précédent, mais insère une adresse explicite dans le message M_i (quand il est non nul) :

$$M_i = \texttt{"Destinataire : 168.0.0.1 Message : xxx... "}$$

Déroulement du tour

1. Chaque utilisateur U_i émet :

   $$\textit{Emission(i)} = \textit{Brouillage(i)} \oplus \textit{Message(i)}$$

   vers un serveur,
2. Le serveur fait le XOR des émissions de tous les utilisateurs et obtient le résultat du tour : $\textit{Résultat} = \bigoplus_{i=1}^{n} \textit{Emission(i)}$

Si le résultat donne lieu à un message (et qu'il n'y a pas de collision), le serveur envoie le message à l'adresse indiquée. En cas de collision, il la notifie aux utilisateurs.

Notons que l'adressage implicite peut aussi être utilisé avec un serveur comme dans le protocole ci-dessus. Dans ce cas le serveur diffuse le résultat du tour à tous les utilisateurs. Comme pour l'utilisation d'un panneau d'affichage pour la diffusion (cf. 1.5.1.2), l'utilisation d'un serveur présente nombreux avantages. Par exemple, le calcul du XOR et la réception de tous les messages l'encombrement réseau qui en découle) n'occupe qu'un seul ordinateur (le serveur) au lieu de celui de chacun des utilisateurs.

[15] Cette méthode de résolution des collisions est analogue à celle utilisée dans les réseaux Ethernet.

1.5.3.2 Justification informelle de la non-observabilité en émission

Pour deux mots binaires S et M, la théorie de l'information nous permet de dire que :

– si on ne connaît rien sur S, on ne peut rien apprendre sur $M \oplus S$ à partir de M.
– si on ne connaît rien sur S, on ne peut rien apprendre sur M à partir de $M \oplus S$.

Ainsi, si les $S_{i,j}$ ne sont utilisés qu'une seule fois, il suffit d'après la première propriété qu'on ne connaisse pas l'un d'entre eux pour qu'on ne puisse rien apprendre sur *Brouillage(i)*. De plus, d'après la deuxième propriété, si on ne sait rien sur *Brouillage(i)* on ne peut rien apprendre sur *Message(i)* à partir de *Diffusion(i)*.

Ces deux remarques nous permettent de dire que pour toute *Diffusion(i)* il suffit *qu'un seul* utilisateur U_j ne trahisse pas U_i en dévoilant $S_{i,j}$ pour qu'on ne puisse rien apprendre sur *Message(i)*. En particulier on ne peut pas savoir si $M_i = 0$ ou pas, c'est-à-dire si U_i a émis un message ou pas.

> *Il suffit donc qu'au moins un utilisateur soit honnête pour que l'anonymat d'émission soit complet.*

Pour que les affirmations précédentes soient vraies il faut absolument que les $S_{i,j}$ ne soient utilisés qu'une fois, et que ceux-ci ne soient pas prévisibles. La proposition de David Chaum était que les utilisateurs échangent des disques optiques avec des grandes quantités de bits aléatoires pour former les $S_{i,j}$. Avec cette solution on peut émettre anonymement autant de bits qu'il y en a dans le disque optique. Dans la pratique, la solution la plus généralement adoptée est que les utilisateurs n'échangent pas des $S_{i,j}$ mais des graines $G_{i,j}$ qui, avec des générateurs pseudo-aléatoires [NIS], permettent de générer un $S_{i,j}$ à chaque tour.

Si on utilise des générateurs pseudo-aléatoires, la sécurité du protocole repose sur la sécurité du générateur et n'est plus inconditionnelle comme dans le protocole de base. Cependant, la simplification apportée par l'utilisation de ces générateurs et le fait que de nos jours la sécurité est souvent basée sur le même type d'hypothèses[16], font que cette dégradation de la sécurité est généralement acceptée.

L'utilisation de l'envoi superposé donne lieu à des groupes d'émission non-observable composés de l'ensemble des utilisateurs participant au protocole, et ceci quel que soit le type d'attaquant.

[16] Ces hypothèses sont que les attaquants ne disposent pas d'une puissance de calcul illimitée, et que sous cette hypothèse, il est possible de générer des nombres pseudo-aléatoires de façon sûre.

1.5.3.3 État de l'art

Les premières références sur l'envoi superposé comme technique d'émission anonyme datent de 1985 et correspondent à une ébauche d'article et une brève référence dans un journal [Cha85], toutes deux écrites par David Chaum. Cette même année Andreas Pfitzmann l'a inclus dans un petit état de l'art [PW85] et en fait une étude extensive dans un rapport interne de l'université Karlsruhe [Pfi85] décrivant comment implémenter des réseaux numériques à intégration de services (RNIS) ayant des propriétés d'anonymat. L'article de David Chaum n'a été publié qu'en 1988 [Cha88]. Il a été suivi d'un ensemble de publications [WP89, Wai89, BdB90] qui ont traité principalement de la détection des utilisateurs provoquant des dénis de service sur le système (les *perturbateurs*). Ce n'est qu'en 2003 qu'une nouvelle publication sur l'envoi superposé a vu le jour [vABH03], traitant de la détection de perturbateurs quand l'émission est faite au sein de très petits groupes anonymes. En 2004 deux nouveaux articles ont étés publiés [YF04, GJ04a], toujours sur la gestion des perturbateurs.

Dans le contexte où nous nous plaçons (cf. 1.6.4), la présence de perturbateurs est peu probable, et en cas de détection d'un perturbateur nous disposons de moyens forts pour le bloquer définitivement. Pour ces raisons nous considérons que le système de pièges de Waidner et Pfitzmann [WP89] suffisent largement à nos besoins.

1.5.3.4 Pratique

De nos jours il n'y a pas d'implémentation aboutie du protocole DC-net (cf. section 2.3), et il n'est pas utilisé à notre connaissance comme moyen pour obtenir un anonymat de communication.

Certains auteurs mentionnent des implémentations du protocole DC-net, mais ces réalisations ont été abandonnées et n'ont abouti à aucune utilisation pratique de nos jours. Nous ne citons que deux de ces travaux ici. Le premier [Dan00] est un projet d'études de l'université de Cambridge et présente une comparaison entre une implémentation de réseaux de MIX (cf. 1.3.3) et une d'un DC-net. Ce travail donne peu de résultats quantitatifs et compare juste les complexités des protocoles de base. Le deuxième travail est une thèse [Mar99] qui fait une étude approfondie des performances des DC-nets, en tenant en compte des améliorations classiques,

pour des groupes allant jusqu'à seize ordinateurs. Dans le chapitre 2, nous faisons une étude de performances des réseaux de type DC-net avec deux différences importantes :

1. nous étudions les groupes allant jusqu'à une centaine d'ordinateurs, ainsi que l'influence que peut apporter une hiérarchisation du réseau sur les performances.
2. nous ne visons pas à remplacer entièrement la pile IP mais uniquement à créer un canal anonyme pour des usages spécifiques.

Dans cette étude, nous comparons nos résultats à ceux de la thèse indiquée ci-dessus.

1.6 Problématique de notre étude

La plupart des publications sur les communications anonymes interactives proposent des systèmes destinés à former des groupes anonymes de grande taille et ouverts à tout utilisateur sur l'ensemble de l'Internet. De nombreuses propositions existent [PPW91, JMP+98, GRS99, FM02, RP04, DMS04, RR98, LS02, DSDa] et on connaît bien les performances qu'on peut atteindre avec ces systèmes. Dans cette section nous allons commenter les limitations intrinsèques aux choix qui ont été faits et énoncer le cadre dans lequel veut s'inscrire notre étude.

1.6.1 Taille des groupes anonymes

De façon évidente, à performances égales, il est préférable d'opérer au sein d'un groupe anonyme de la plus grande taille possible. Malheureusement, de nos jours, les solutions existantes ou proposées dans la littérature pour des groupes anonymes importants offrent des performances qui sont très inférieures à celles des protocoles ne fournissant pas d'anonymat. Ceci a pour conséquence que certains utilisateurs se tournent vers des protocoles moins surs [ANZ], d'autres n'utilisent des protocoles de communication anonyme que de façon sporadique, et la grande majorité des utilisateurs n'en utilise jamais.

Les raisons qui font que les performances ne sont pas à la hauteur des attentes des utilisateurs sont diverses. Dans certains cas ceci est dû intrinsèquement au faible

facteur d'échelle supporté par le protocole comme c'est le cas pour les solutions basées sur l'envoi superposé 1.5.3. Dans d'autres cas, c'est dû à la complexité du système d'un point de vue global, comme pour les réseaux basés sur des structures de pair à pair [FM02, RP04].

De rares systèmes supportent un facteur d'échelle important comme JAP [DSDa] ou Tor [DMS04], et sont basés sur des architectures simples et performantes. Mais ces systèmes sont confrontés à un autre problème, celui du partage des ressources. En effet, dans ces systèmes, le nombre de serveurs fournissant le support pour les communications anonymes est limité[17] parce que jusqu'à maintenant toute tentative de faire payer les services d'anonymisation à échoué, et donc plus il y a d'utilisateurs connectés au service, plus les performances baissent.

La recherche de solutions efficaces et à faible coût qui permettraient d'avoir des communications anonymes avec de grands groupes, donne lieu chaque année à un nombre important d'articles. Cependant, nous pensons que certains domaines d'application pour les communications anonymes sont trop négligés. Ainsi il n'existe aucune implémentation aboutie dédiée à l'établissement de communications anonymes au sein d'un groupe restreint, et ce n'est que récemment que quelques articles [YF04, vABH03] se sont intéressés à l'émission au sein de petits groupes anonymes.

1.6.2 Les groupes ouverts et les groupes fermés

Il est avantageux de disposer de solutions permettant d'avoir des groupes anonymes tels qu'en principe, il soit possible à n'importe quel utilisateur dans le monde de s'y intégrer. Les solutions disposant de tels *groupes ouverts*, ont l'avantage d'augmenter sensiblement le nombre d'utilisateurs potentiels et par là-même, permettent la formation de groupes anonymes de grande taille.

Dans ces groupes ouverts, plusieurs problèmes se présentent. Premièrement, la confiance que l'on peut attribuer aux utilisateurs est extrêmement faible, et deuxièmement on dispose de peu de moyens pour prendre des actions contre ceux-ci.

[17] Dans le cas de Tor, le nombre de serveurs augmente avec le volontariat qui est directement lié au nombre de clients. Cependant ceci conduit à réduire la proportion de serveurs de qualité professionnelle entraînant donc une diminution des performances, ce qui est d'ailleurs observable au niveau de la variance dans les débits et dans la latence du service.

En conséquence, toutes les solutions qui ont offert un service de communications anonymes ouvert à n'importe quel utilisateur ont été confrontées à des usages abusifs de leur service, ce qui leur a valu dans certains cas des problèmes avec la justice [DSDb], conduisant parfois même à la fermeture du service [PNT].

De plus, le manque de confiance et la difficulté à prendre des mesures efficaces contre les actions répréhensibles de ces utilisateurs font que souvent on se voit obligé de tenir compte de certaines attaques actives.

Ainsi, par exemple dans le cas des DC-net, les perturbateurs peuvent opérer vis-à-vis d'un FAC qui dispose de peu de moyens contre eux et qui se voit donc obligé de mettre en place des moyens rapides de découverte des perturbateurs. Dans le cas des MIX, accepter des messages de n'importe quel utilisateur, ce qui est commun dans les FAC ayant des groupes ouverts, rend les attaques par inondation plus simples [BPS00]. Ces attaques peuvent en effet être facilement réalisées par des attaquants contrôlant beaucoup d'ordinateurs[18] et se faisant passer pour un grand nombre d'utilisateurs.

Par ailleurs, les solutions proposant des groupes ouverts, et donc changeants, sont plus sensibles aux attaques par intersection décrits dans la section 1.4 et ne peuvent donc garantir un service d'anonymat face aux observateurs globaux, ou aux attaquants suspicieux.

En revanche, dans le cas des groupes fermés, les attaques par intersection sont beaucoup moins efficaces, et une authentification des utilisateurs peut être mise en place pour éviter les attaques par inondation. Ces groupes étant moins changeants, l'exclusion d'un attaquant donne plus de répit au système que dans le cas des groupes ouverts et on peut donc se permettre d'utiliser des moyens de découverte des perturbateurs moins rapides et souvent beaucoup moins coûteux. Enfin, dans les groupes fermés, la confiance que l'on peut attribuer aux autres utilisateurs est souvent plus importante, et les mécanismes de réputation sont beaucoup plus faciles à implémenter.

1.6.3 L'Internet et les réseaux locaux

Un grand nombre d'applications sont beaucoup plus intéressantes si les utilisateurs peuvent être distribués sur l'ensemble de l'Internet. Par exemple un système

[18] Par exemple, un *botnet de zombies* [Gee05].

de voix sur IP ou d'échange de fichiers anonymisé à l'échelle globale est autrement plus intéressant que le même système limité à un réseau local. Une telle distribution est d'ailleurs le seul moyen pour obtenir des groupes anonymes ayant plus que quelques centaines de membres, et rend difficile l'analyse globale du trafic. Cependant, il faut bien comprendre que ceci implique de renoncer à de nombreux avantages.

Dans un réseau local nous disposons de façon immédiate d'un support communication extrêmement performant : la bande passante disponible ainsi que la faible latence du réseau permettent de considérer des applications qui ne peuvent pas être mises en place sur l'Internet. Par ailleurs, la possibilité de réaliser des diffusions ainsi que l'utilisation généralisée des réseaux virtuels permettent l'implémentation de protocoles qui ont été abandonnés dans les réseaux à grande échelle. De plus, le contrôle sur le support et les serveurs pouvant fournir les services d'anonymat permettent d'adapter le service aux besoins réels des utilisateurs. Enfin, dans un réseau local un attaquant sera facilement associé à une personne physique, et risquera de souffrir de conséquences graves pour ses actes : licenciement, poursuites, etc. Les attaquants sont limités par ces faits et en général ceci permet de considérer des modèles d'attaquants moins contraignants et d'optimiser les performances des protocoles.

1.6.4 Cadre de travail

La problématique de cette thèse est d'obtenir des services fournissant un anonymat de communication à faible latence et sans tiers de confiance. Pour cela nous avons décidé d'étudier quelles performances peuvent être obtenues en bande passante et en latence, et quelles approches sont les meilleures, quand on introduit certaines contraintes sur les groupes d'utilisateurs potentiels du service fournissant un anonymat de communication. En particulier, nous nous intéressons aux groupes fermés, aux réseaux locaux, et aux groupes anonymes de petite taille. Les communautés d'intérêt sur l'Internet, les réseaux locaux privés, et les entreprises sont quelques exemples de cadres dans lesquels nos résultats pourraient se voir appliqués.

1.6.4.1 Modèles d'attaquant

Les hypothèses qui sont plausibles dans ce contexte, ainsi que les moyens à notre disposition et les applications envisageables diffèrent fortement de celles du contexte traditionnel. En particulier, nous nous voyons forcés de prendre en compte deux modèles d'attaquant extrêmement forts. Le premier concerne uniquement les réseaux locaux : l'administrateur du réseau, qui peut en général accéder à l'ensemble des ordinateurs connectés, peut être considéré comme un attaquant potentiel. Le deuxième modèle d'attaquant est l'observateur global, qui est d'autant plus plausible que nous nous éloignons du contexte des grand groupes ouverts sur l'ensemble de l'Internet. En effet, il est plus facile d'observer globalement un groupe s'il change rarement, s'il est petit, ou s'il est dans un réseau local.

1.6.4.2 Approches possibles

Suite aux arguments présentés dans la section 1.5.2, nous considérons que le cadre des groupes d'utilisateurs sous contraintes se prête mal à l'utilisation de relais. Dans un cadre plus général, il n'est peut-être pas impossible de réaliser des systèmes permettant des communications anonymes à faible latence avec des relais. Cependant il faut constater qu'en vingt ans de recherche, malgré les grands progrès réalisés, la plupart des applications hautement interactives ne peuvent pas être utilisées avec les réseaux de relais distribués sur Internet.

Dans le reste de cette étude, nous ne considérons que les solutions qui résistent aux attaques par des observateurs globaux et qui ne nécessitent pas la confiance des utilisateurs. Pour cela, il faut que le système utilisé permette aux utilisateurs d'émettre et de recevoir de façon non-observable.

Pour former des groupes d'émission non-observable, il ne faut pas pouvoir distinguer si un utilisateur émet un message ou pas. Les utilisateurs peuvent donc soit utiliser du bourrage chiffré soit émettre en suivant un protocole d'envoi superposé. Pour former des groupes de réception non-observable, la seule option a priori possible est d'utiliser la diffusion et l'adressage implicite.

Ceci donne lieu à uniquement deux possibilités pour implémenter des systèmes fournissant un anonymat de communication quand on considère des attaquants globaux ou suspicieux. La première est d'utiliser du bourrage chiffré et la diffusion avec adressage implicite (cf. 1.5.1), et la deuxième d'utiliser l'envoi superposé et

la diffusion avec adressage implicite (cf. 1.5.3).

1.6.4.3 Utilisation de serveurs

Autant si on utilise le bourrage chiffré que si on utilise l'envoi superposé, le service fournissant un anonymat de communication peut être implémenté de façon distribuée ou par un serveur centralisé recueillant les informations et les diffusant pour les utilisateurs. L'utilisation de tels serveurs entraîne souvent des meilleures performances et simplifie les protocoles (cf. 1.5.1.2 1.5.3.1). C'est pourquoi, même si dans certains cas une implémentation distribuée peut être intéressante, nous avons décidé de nous concentrer sur les implémentations centralisées. Si on utilise du bourrage chiffré, on parle d'un panneau d'affichage chiffré (EBBS pour *Encrypted Bulletin Board System*), en revanche, si on utilise l'envoi superposé, on parle d'un serveur DC-net.

Dans le chapitre suivant nous réalisons une étude de performances de ces deux approches. Dans le chapitre 3 nous présentons une primitive cryptographique nommée la récupération d'informations privée (*Private Information Retrieval* ou PIR) qu'on utilise dans le chapitre 4 pour proposer des alternatives aux EBBS et serveurs DC-net.

Chapitre 2

Solutions basées sur la diffusion avec adressage implicite

Dans ce chapitre, nous analysons dans le contexte des groupes anonymes sous contraintes plus en détail deux techniques permettant la diffusion avec adressage implicite : les panneaux d'affichage chiffrés (ou EBBS, pour Encrypted Bulletin-Board-System) et les serveurs DC-net. Pour les premiers, leur simplicité fait qu'une implémentation n'est pas nécessaire et nous nous contentons que de présenter dans la section suivante une évaluation théorique de leurs performances. Pour les seconds, il est nécessaire de réaliser une implémentation partielle et de l'étudier sous divers paramètres pour pouvoir évaluer quel est l'ordre de grandeur des performances auxquelles on peut s'attendre. Les sections 2.2 et suivantes présentent ce travail.

2.1 Évaluation des performances d'un EBBS

Quoi que plusieurs approches soient possibles, un EBBS (voir figure 2.1) peut être modélisé de la façon suivante :

1. à chaque tour, chaque utilisateur connecté écrit un message de taille fixe dans un emplacement qui lui est réservé au sein de l'EBBS,
2. après réception de ces messages, l'EBBS diffuse le contenu de l'ensemble des emplacements, à tous les utilisateurs connectés.

Le coût de traitement pour le serveur est pratiquement nul, et sa seule limitation est la bande passante dont il dispose en émission et en réception. D'un autre côté, la taille de l'ensemble des messages peut être extrêmement importante et ainsi il faut considérer aussi les limitations en bande passante en réception des utilisateurs.

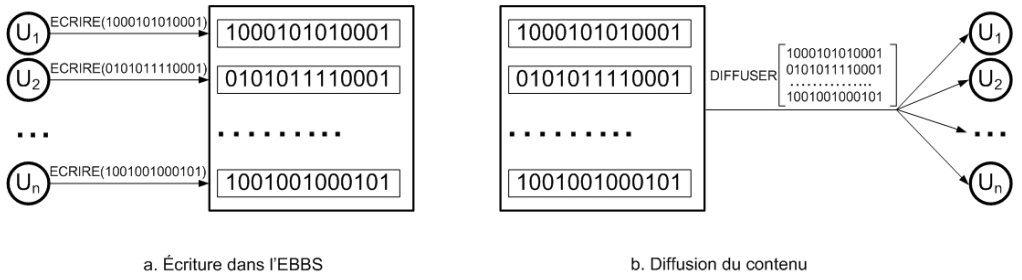


FIG. 2.1: Tour d'émission avec un EBBS.

Dans un réseau local, pour attribuer une bande passante B à chaque utilisateur, un serveur doit pouvoir émettre en diffusion et recevoir avec une bande passante $n \times B$, n étant le nombre d'utilisateurs connectés à l'EBBS. Si les utilisateurs ne se trouvent pas dans un même réseau ou s'il n'est pas possible pour le serveur d'émettre en diffusion, celui-ci doit transmettre en point-à-point l'ensemble des messages à chacun des utilisateurs et dans ce cas il lui faut une bande passante $n \times B$ en réception et $n^2 \times B$ en émission. Quant aux utilisateurs, ils émettent en permanence soit des informations utiles, soit du bourrage chiffré, les communications avec l'EBBS occupent donc une bande passante B en émission et $n \times B$ en réception.

2.1.1 Utilisation de tours d'appel

L'utilisation d'adresses implicites peut entraîner un coût de traitement important. En effet, si chaque utilisateur doit essayer de déchiffrer tous les messages diffusés ceci devient très vite le facteur limitant quand le nombre d'utilisateurs connectés augmente. On peut fortement réduire le coût de traitement en instaurant ce que nous appellerons des *tours d'appel*, c'est-à-dire des tours dédiés uniquement à initialiser des communications. Les utilisateurs ne déchiffrent l'ensemble des messages reçus que lors des tours d'appel, qui peuvent avoir une période assez grande (quelques secondes par exemple).

Ces tours peuvent être implémentés de deux façons, par l'utilisation de bourrage chiffré, ou par envoi superposé. Dans le cas de l'utilisation bourrage chiffré, ces tours seront appelés des tours d'appel de type EBBS. Dans le cas de l'envoi superposé on les appellera des tours d'appel de type DC-net.

Les tours d'appel de type EBBS se déroulent exactement de la même façon que les tours de communication dans un EBBS : chaque utilisateur envoie un message d'appel chiffré ou un message de bourrage à l'EBBS, puis l'ensemble des messages est diffusé. Étant donné qu'uniquement les utilisateurs connectés à l'EBBS essayent de déchiffrer les messages d'un tour d'appel, il est important de remarquer que cette approche ne permet pas de contacter anonymement des utilisateurs extérieurs.

Dans les tours d'appel de type DC-net, les utilisateurs agissent comme avec un serveur DC-net : à chaque tour d'appel, chaque utilisateur émet en point-à-point vers le serveur un message brouillé. Le résultat du tour est soit nul si aucune nouvelle connexion n'est demandée, soit un message chiffré pour l'interpellé si une nouvelle connexion doit être établie, soit encore du brouillage s'il y a une collision (cf. 1.5.3). Si l'interpellé est déjà membre du groupe d'anonymat, le message chiffré peut utiliser de l'adressage implicite. En utilisant un adressage explicite ce type de tour d'appel permet aussi d'établir une connexion vers un destinataire qui n'est pas encore membre du groupe d'anonymat, avantage important par rapport aux tours d'appel de type EBBS. D'un autre côté, avec un protocole d'envoi superposé, les utilisateurs comme les administrateurs de l'EBBS peuvent savoir quand le résultat du tour donne lieu à un appel et donc quand une communication commence ce qui peut être une information importante en soi pour un attaquant. Nous présentons plus loin dans ce chapitre (cf. 2.2.2.3) des techniques permettant d'occulter partiellement cette information auprès des utilisateurs.

Quelque soit le type de tour d'appel, un utilisateur déchiffrant lors d'un tour d'appel un message lui indiquant une communication entrante dans un emplacement donné, ne déchiffrera à chaque tour de communication que les messages correspondant à cet emplacement (en plus de l'ensemble des messages à chaque tour d'appel). Un utilisateur ne communicant pas ne déchiffre que les messages des tours d'appel, de telle sorte que le coût de traitement pourra être négligé dans la plupart des applications.

2.1.2 Performances

Étant donné que pour la plupart des applications, le coût de traitement n'est pas un facteur limitant, les performances d'un EBBS dépendent principalement de la bande passante dont disposent le serveur et les utilisateurs. Nous distinguons trois situations qui de nos jours couvrent la plupart des utilisateurs :

– les utilisateurs dans des réseaux locaux, typiquement avec une bande passante de 100 Mbits/s,
– les utilisateurs ayant une connexion ADSL, pour laquelle on prend comme valeur de référence 1 Mbit/s en réception et 128 Kbits/s en émission,
– les utilisateurs ayant une connexion mobile de type UMTS, qui de nos jours donne lieu à une bande passante de 64 Kbits/s en émission et 384 Kbits/s en réception.

Pour fournir un service d'anonymat de communication, une part de la bande passante de l'utilisateur doit être réservée pendant des laps de temps importants. Nous limitons donc cette part de bande passante à une fraction du total dont dispose l'utilisateur. Dans le cas des réseaux locaux nous fixons cette limite pour chaque utilisateur à 1 Mbit/s en émission et en réception. Dans les autres cas nous fixons la limite à dix pour cent de la bande passante disponible. Pour le serveur nous considérons qu'il est dédié, et donc qu'une occupation à cent pour cent est possible. Dans le cas d'un réseau local, les serveurs se trouvent généralement sur des dorsales, et peuvent disposer de nos jours d'une bande passante de 1 Gbit/s en émission et en réception. Si le serveur doit envoyer et recevoir les données à travers l'Internet on fixe cette limite soit à 10 Mbits/s, correspondant à un coût en dessous d'une centaine d'euros par mois avec un trafic total mensuel illimité, soit à 100 Mbits/s pour un coût d'environ cinq cents euros par mois.

Souvent dans ce mémoire nous utiliserons l'exemple de la voix sur IP pour l'évaluation des serveurs, car c'est une application très en vogue de nos jours, qu'il est difficile d'implémenter anonymement avec des groupes anonymes sans contraintes. Si on suppose qu'il suffit d'avoir $B = 10$ Kbits/s pour qu'un utilisateur puisse communiquer par voix sur IP on peut déduire combien d'utilisateurs peuvent former un groupe anonyme de communication dans les différentes situations.

Dans le cas des réseaux locaux la limitation est la bande passante en réception des utilisateurs qui fait que la taille maximale du groupe anonyme de communication est d'environ cent utilisateurs. Dans le cas des utilisateurs connectés à travers l'Internet, un serveur ayant une bande passante de 10 Mbits/s ne peut gérer

plus d'une trentaine de clients. Un serveur disposant d'une bande passante de 100 Mbits/s quant à lui peut diffuser les informations à travers l'Internet à des groupes ayant une taille allant jusqu'à une centaine d'utilisateurs environ. Du côté des utilisateurs, si ceux-ci sont connectés par une liaison ADSL à 1 Mbit/s la taille maximale du groupe est de dix utilisateurs ; si ceux-ci disposent d'une liaison ADSL de deuxième génération à 5 Mbits/s cette limite peut être poussée jusqu'à cinquante utilisateurs. Dans le cas de l'UMTS la taille maximale est de trois utilisateurs (en admettant que quinze pour cent de la bande passante en émission soit occupée).

Ce serveur peut donc fournir un service de voix sur IP anonyme dans les réseaux locaux de taille moyenne. Dans le cas où les utilisateurs sont distribués sur l'Internet, cette approche peut se révéler coûteuse, et elle est en général restreinte à des groupes anonymes de petite taille. Enfin, elle ne semble pas utilisable pour des utilisateurs ayant des connexions de type UMTS. Pour la vidéo-conférence sur IP où on supposera qu'un débit de 128 Kbits/s est suffisant, une telle approche ne peut pas être adoptée pour des utilisateurs distribués sur l'Internet, et elle est pratiquement inutile (maximum de huit utilisateurs) dans le cas des réseaux locaux.

Un autre désavantage de l'EBBS est qu'une collusion entre un utilisateur interpellé et les administrateurs du serveur lève l'anonymat sur l'interpellant. En revanche, ni les administrateurs ni les utilisateurs ne peuvent connaître le nombre de communications qui ont lieu à un instant donné, et tous les utilisateurs peuvent communiquer simultanément. Cette approche est donc particulièrement adaptée aux applications pour lesquelles le nombre de communications concurrentes peut être élevé.

2.2 Serveurs DC-net

Dans le reste de ce chapitre nous présentons une implémentation partielle d'un serveur DC-net ainsi qu'une étude des performances de cette implémentation. Notre but n'est pas de présenter un système complet mais de donner une idée raisonnable de l'ordre de grandeur des performances que peut atteindre un serveur DC-net en fonction du nombre d'utilisateurs connectés. David Martin donne des résultats de performance dans son étude sur l'envoi superposé [Mar99], cependant ses tests correspondent à l'étude d'une implémentation où l'envoi superposé est intégré au sein de la pile IP avec un certain nombre de choix d'implémentation qui font que les résultats donnés ne sont pas caractéristiques de l'utilisation de l'envoi su-

perposé en soi, mais plutôt de l'implémentation qu'il réalise. Au contraire, nous avons décidé de réaliser des tests sur une implémentation qui ne s'occupe de faire que l'indispensable pour tout serveur DC-net : génération des messages par les clients puis XOR au niveau du serveur et diffusion. Deuxièmement, David Martin réalise des tests dans un réseau de très petite taille alors que dans ce chapitre nous réalisons les tests pour un grand nombre d'ordinateurs sur un réseau d'entreprise réel, et nous étudions l'impact de l'utilisation d'un tel réseau sur les performances des DC-nets [Agu06b].

Dans un deuxième temps nous étudions l'impact de la distribution des calculs sur plusieurs serveurs mandataires. L'utilisation de mandataires permet de partitionner l'ensemble des utilisateurs en groupes, chacun connecté à un mandataire, les mandataires étant eux-mêmes connectés au serveur DC-net. Les mandataires font le XOR des messages reçus des membres de leur groupe, et envoient le résultat au serveur qui ne doit dès lors faire que le XOR des messages issus de chacun des groupes. L'utilisation de mandataires dans le cadre de l'envoi superposé n'est pas une idée nouvelle. Par exemple, David Martin utilise des mandataires dans son implémentation de l'envoi superposé, et certains systèmes sont entièrement basés sur l'utilisation des hiérarchies de mandataires [DO00, SBS02].

L'utilisation de mandataires présente deux avantages. Premièrement, étant donné que l'ensemble des messages envoyés à un mandataire par les utilisateurs qui sont connectés à celui-ci donne lieu à un unique message en sortie par tour, une réduction importante du coût des communications peut être obtenue si ces mandataires sont bien placés. Deuxièmement, la latence introduite pour une communication avec un serveur DC-net unique est proportionnelle au nombre d'utilisateurs connectés, n. Si ces utilisateurs sont fractionnés en $\sqrt{n}$ groupes de taille $\sqrt{n}$, chacun de ces groupes étant connecté à un mandataire, et si on peut négliger la latence du réseau, la latence introduite sera $2 \times \sqrt{n}$. Si nous organisons les mandataires en un arbre binaire où les utilisateurs sont les feuilles de l'arbre, ceci donne théoriquement une latence proportionnelle uniquement au nombre de mandataires qu'il faut traverser pour aller d'une feuille jusqu'à la racine, puisque dans cette configuration chaque mandataire a toujours deux clients, indépendemment de n. La profondeur d'un arbre binaire étant $log(n)$ on en déduit qu'en principe l'utilisation de mandataires permet d'atteindre une latence logarithmique vis-à-vis du nombre d'utilisateurs, tant que la latence introduite par le réseau est négligeable. Nous ne connaissons aucune étude de l'impact sur les performances de l'utilisation d'arbres de mandataires et dans la section 2.2.3.4 nous étudions à quel point

on peut s'approcher de la limite logarithmique théorique.

2.2.1 L'implémentation d'un serveur DC-net

L'utilisation de fils d'exécution (*threads*) distincts pour l'émission et la réception, et d'un fil d'exécution par client au niveau du serveur est la première approche que nous avons essayée. Nous avons finalement abandonné ce choix au niveau du serveur pour plusieurs raisons. Dans le cas des fils d'exécution concurrents, les changements de contexte sont réalisés de façon indépendante du protocole d'envoi superposé, à moins qu'un système de signalisations ne soit mis en place pour forcer l'ordonnanceur à donner la priorité aux fils d'exécution n'ayant pas encore participé au tour, ou au fil d'exécution de diffusion si le tour est fini. Dans le cas des fils d'exécution collaboratifs, ceux-ci doivent décider par eux-mêmes à quel fil d'exécution ils laissent la main , le meilleur choix étant de passer la main à un fil d'exécution qui a un message en attente s'il y en a, et à un fil d'exécution qui n'a pas encore reçu de message pendant le tour, dans le cas contraire. Quand tous les fils d'exécution s'occupant des clients ont ont été exécutés, le dernier doit passer la main à celui qui réalise la diffusion. On constate que la mise en place de la signalisation pour les fils d'exécution concurrents, et la mise en place de mécanismes permettant aux fils d'exécution collaboratifs de décider à qui passer la main sont très coûteuses par rapport à la tache principale par tour des fils d'exécution, qui est de recevoir un message et de réaliser un XOR.

Dans le cas des fils d'exécution concurrents, nous avons éliminé la totalité des verrous afin de réduire le coût induit par les mécanismes de synchronisation. Nous avons aussi instauré un système de récompenses afin d'auto-réguler les temps de sommeil entre les fils d'exécution pour que le serveur soit aussi efficace avec peu qu'avec beaucoup de clients. Dans le cas des fils d'exécution collaboratifs nous avons réalisé des tests avec un enchaînement séquentiel des fils d'exécution, ce qui ne demande aucune ressource mais ne permet pas de choisir ceux-ci de façon optimale, puis des tests avec des notifications et des tableaux aidant les fils d'exécution à choisir leur successeur. Dans tous les cas les performances ont été très inférieures à celles que nous avons obtenu en travaillant en mono-tâche.

2.2.1.1 Les clients

Les clients que nous avons implémentés sont composés de deux fils d'exécution. Le premier reçoit en UDP les messages diffusés par le serveur et maintient un compteur *nbmr* déterminant quel est le tour de diffusion reçu le plus avancé. Le deuxième fil d'exécution émet en TCP les messages brouillés nécessaires à la complétion des tours d'envoi superposé. Ce fil d'exécution dépend d'un paramètre *b* lui indiquant combien de tours d'avance il peut avoir par rapport au compteur *nbmr*. Ce paramètre *b* correspond donc à la taille de tampon des clients et il a une influence très forte sur la bande passante que peut fournir le serveur ainsi que sur la latence. Pour des trop faibles valeurs de *b* la bande passante que le serveur DC-net peut fournir aux utilisateurs chute, tandis que la latence augmente rapidement avec *b*. La valeur optimale de *b* dépend d'un grand nombre de paramètres. Pour nos tests, nous avons choisi *b* par expérimentation.

2.2.1.2 Le serveur

Le serveur utilise des foncions de type *select()* et *poll()*, qui permettent à un seul fil d'exécution de gérer les messages des clients séquentiellement au fur et à mesure que ceux-ci arrivent au serveur. Afin d'optimiser le performances, le serveur dispose lui même d'un tampon de taille *b* (cf. section précédente), qui lui permet de traiter les messages sur plusieurs tours en parallèle si des clients sont en retard pour certains tours. Quand un tour est terminé, le serveur diffuse simplement le résultat du XOR.

Les mandataires sont en tout point similaires au serveur, excepté le fait que lorsqu'un tour est complété, ils ré-émettent en point-à-point le résultat vers le serveur DC-net, ou vers le mandataire de niveau immédiatement supérieur au lieu de le diffuser.

2.2.2 Protocoles non implémentés

Pour une utilisation pratique du serveur DC-net certains protocoles auxiliaires doivent être implémentés. Par exemple, un protocole de gestion des secrets communs permettant de générer les brouillages et un protocole de gestion des connexions et déconnexions sont indispensables au fonctionnement pratique d'un tel serveur.

Nous avons décidé de privilégier l'expérimentation et la généralité des résultats, et pour cette raison nous nous sommes restreints à réaliser des tests pour des clients et des serveurs ne réalisant que les opérations correspondant au déroulement des tours d'envoi superposé.

Cependant, même si les résultats de nos tests ne portent que sur une telle expérimentation, nous présentons ici trois protocoles auxiliaires qui dans une implémentation réelle sont à nos yeux d'une grande utilité.

Le premier permet de grouper les tours par communication et donne lieu à une réservation de canaux, qu'on peut assimiler aux emplacements réservés aux utilisateurs que nous avons présentés dans l'EBBS. Présupposer l'utilisation de ce protocole simplifie donc les comparaisons entre les deux approches, et permet de plus d'affirmer que les collisions n'ont pratiquement aucun impact sur les performances du système. Le deuxième protocole permet de doubler la bande passante dont disposent les utilisateurs lors des communications bidirectionnelles et doit donc être pris en compte pour toute comparaison avec les autres approches. Ce même protocole, sous une forme plus générale, permet aussi de résoudre un nombre quelconque de collisions de façon extrêmement simple et efficace, et nous permet plus en avant dans ce mémoire d'utiliser le serveur DC-net comme une primitive utile dans de nombreuses occasions. Enfin, le troisième protocole, d'une grande simplicité, permet d'occulter le nombre de communications en cours aux utilisateurs au prix d'une augmentation significative du coût en communication.

2.2.2.1 La réservation de canaux

Lors d'un tour d'envoi superposé, n'importe quel utilisateur peut émettre un message. Si plusieurs utilisateurs communiquent simultanément, leurs émissions seront mélangées et dans beaucoup de cas le nombre de collisions sera important. Pour éviter cela, nous pouvons exécuter en parallèle plusieurs groupes de tours. Nous appelons chacun de ces groupes un canal. Un de ces canaux est ce que nous appelons le canal de contrôle. Les tours dans le canal de contrôle, ou *tours de contrôle*, sont espacés régulièrement, par exemple de quelques secondes, et servent à créer des *canaux de communication*. Quand un utilisateur souhaite initialiser une communication, il commence par envoyer un message C lors d'un tour de contrôle en demandant la création d'un canal de communication. Quand le résultat d'un tour de contrôle révèle une demande de création de canal, tous les utilisateurs commencent un ensemble de tours d'envoi superposé qui composent

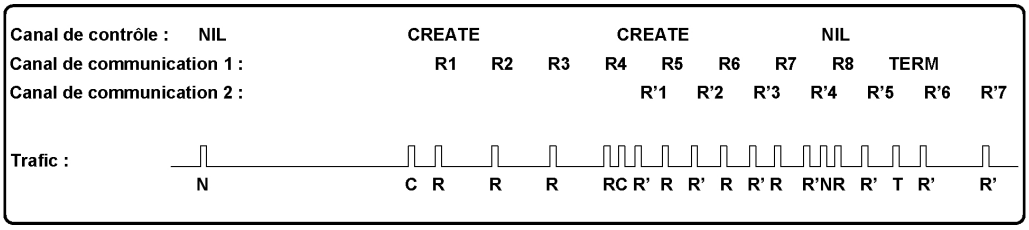


FIG. 2.2: Réservation de canaux.

le canal de communication. Ces tours sont indépendants des tours de contrôle, et se font à une fréquence plus élevée, la fréquence de communication. Dans un canal de communication, seul l'utilisateur ayant demandé sa création transmet des informations, et par conséquent il n'y aura pas de collision. Quand l'utilisateur finit de transmettre, il envoie lors d'un tour de contrôle un message T demandant la fermeture du canal de communication, et les utilisateurs arrêtent d'exécuter les tours d'envoi superposé correspondant à ce canal (voir figure 2.2). Des collisions sont possibles lors des tours de contrôle, mais comme les requêtes de création de canal sont, pour la plupart des applications, beaucoup moins fréquentes que l'émission de messages, le nombre de collisions à résoudre est grandement réduit.

2.2.2.2 La réception superposée

Quand deux utilisateurs A et B communiquent à travers un canal, ils peuvent volontairement transmettre simultanément au sein de celui-ci, de telle façon qu'à chaque tour d'envoi superposé il y ait une collision entre leurs messages. Ainsi, si lors d'un tour leurs messages respectifs sont M_A et M_B, le résultat du tour d'émission sera $M_A \oplus M_B$. Chacun des deux utilisateurs peut obtenir le message de l'autre à partir de ce résultat en réalisant un XOR entre celui-ci et leur message. Par exemple, A obtient $M_A \oplus M_B \oplus M_A = M_B$. Ainsi, avec l'utilisation de l'envoi superposé les canaux de communication sont duplex, alors qu'avec un EBBS chacun des emplacements ne peut servir que pour des transmissions unidirectionnelles.

La réception superposée peut être généralisée pour des collisions d'un nombre quelconque d'utilisateurs [Pfi90, Wai89]. Par l'utilisation de moyennes ce protocole permet de résoudre une collision de n utilisateurs en n tours d'envoi superposé, avec un coût négligeable en communication et en traitement. Il peut être très intéressant d'utiliser ce protocole pour les tours de contrôle dans le protocole de

réservation de canaux.

2.2.2.3 Occulter le nombre de communications

La connaissance du nombre m de communications en cours est dans beaucoup de situations une donnée sensible, d'autant plus que cette valeur indique non seulement le nombre de communications mais aussi quand une communications débute (quand m est incrémenté) et finit (quand m est décrémenté). Avec un serveur DC-net autant les administrateurs de celui-ci que les utilisateurs connaissent m. Si occulter m auprès des administrateurs du serveur semble un problème difficile, le faire auprès des utilisateurs est simple mais coûteux d'un point de vue des communications. En effet, pour ce faire, il suffit que le serveur simule un nombre constant de communications, et que les tours de contrôle ne soient pas diffusés. Par exemple, un tel serveur fournissant un service de voix sur IP anonymisé peut simuler que cinq communications ont constamment lieu, et refuser qu'il y en ait plus que cinq à tout moment. Si le nombre de communications qui ont réellement lieu est inférieur, les résultats des tours d'envoi superposé des canaux non utilisés sont systématiquement nuls et le serveur les remplace par du bourrage chiffré.

Avec un tel système, la bande passante occupée en émission et en réception par les utilisateurs est en permanence $m_{MAX} \times B$, m_{MAX} étant le nombre maximum de communications que le serveur peut occulter. En faisant varier m_{MAX}, un compromis peut être trouvé entre le serveur DC-net traditionnel où les utilisateurs occupent $m \times B$ en émission et en réception et un EBBS où ils occupent $n \times B$ en réception. Bien sûr, l'envoi superposé rend le système plus complexe et l'occupation en émission peut être rapidement un problème, ce pourquoi ce système est uniquement envisageable si $m_{MAX} \ll n$.

Même dans ce cas, un utilisateur peut obtenir des renseignements sur m en démarrant des fausses communications. Ainsi, par exemple, si le serveur refuse d'initialiser sa communication, l'utilisateur peut conclure que $m = m_{MAX}$. Donc, cette occultation de m est contournable. Cependant, si les utilisateurs peuvent apprendre la valeur de m, ils auront quand même beaucoup moins de précision sur son évolution quand sa valeur est occultée, spécialement si les tours de contrôle sont faits à des intervalles de temps proches de la durée d'une communication. De plus, cette attaque est active, et donc détectable, alors que dans le cas d'un serveur normal, la simple observation de m est une attaque indétectable. Si l'attaquant veut réaliser plus de quelques observations isolées de m, cette l'attaque sera manifeste ce qui

peut entraîner la mise en place de pièges (cf. 1.5.3.3) et de sanctions (cf. 1.6.2).

2.2.3 Les tests réalisés

Nous avons évalué la bande passante disponible, à partager entre l'ensemble des utilisateurs. Cette bande passante est brute, on ne considère pas l'existence d'entêtes, de correction d'erreurs, ou d'encapsulation des données, puisque ces considérations sont spécifiques à chaque implémentation réelle. La latence donnée est la latence maximale observée pour des ensembles de quelques dizaines de milliers de paquets en excluant un pour mille des latences les plus importantes (afin de lisser les courbes). On appelle le résultat d'un tour d'envoi superposé une cellule. L'ensemble de nos test ont étés réalisés pour trois tailles de cellule : un, huit et soixante-quatre Kbits. Les courbes pleines dans les schémas représentent les interpolations que la théorie nous permet de faire et les courbes en pointillés les résultats obtenus par expérimentation. Autant dans la figure 2.4 que dans la figure 2.6, l'interpolation des deux premiers schémas est la même : nous avons séparé ces deux schémas afin de rendre les résultats plus visibles, mais ils correspondent à un même ensemble de tests.

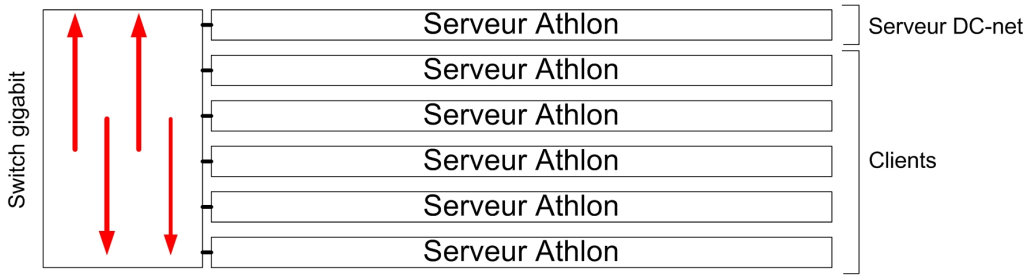


FIG. 2.3: La grappe d'ordinateurs.

Deux ensembles de tests on été réalisés : l'un dans une grappe d'ordinateurs et l'autre dans un réseau local complexe et de grande taille. Nous avons choisi cette approche pour avoir dans le premier cas des résultats dans un contexte idéal, avec un seul switch de grande puissance, avec virtuellement aucun trafic mis à part celui généré par l'envoi superposé et pour un ensemble homogène de clients, et dans le second cas pour pouvoir observer la dégradation des performances quand on se place dans un réseau local réel et complexe. On suppose que les résultats

pour des réseaux locaux de complexité intermédiaire peuvent se déduire de nos résultats par interpolation à partir des deux extrêmes que nous présentons.

2.2.3.1 La grappe

Le premier ensemble de tests a été réalisé sur une grappe de six serveurs Athlon placés derrière un switch Gigabit (voir figure 2.3). Le premier élément de la grappe exécute le serveur DC-net, et les autres cinq éléments exécutent plusieurs clients chacun.

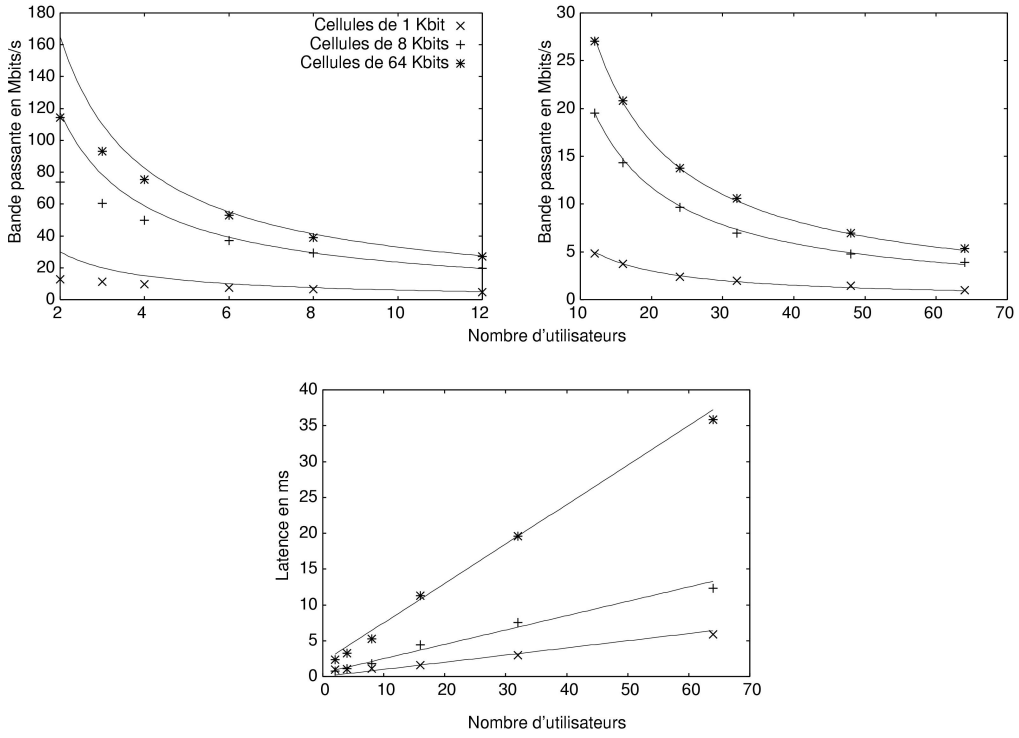


FIG. 2.4: Performances dans la grappe.

La figure 2.4 donne les résultats de ces tests en bande passante et en latence. Comme le serveur partage sa puissance de calcul entre les clients, la décroissance de la bande passante est en $O(1/n)$ et la latence augmente linéairement, ce qui correspond aux résultats attendus. Il y a une petite différence entre l'interpolation

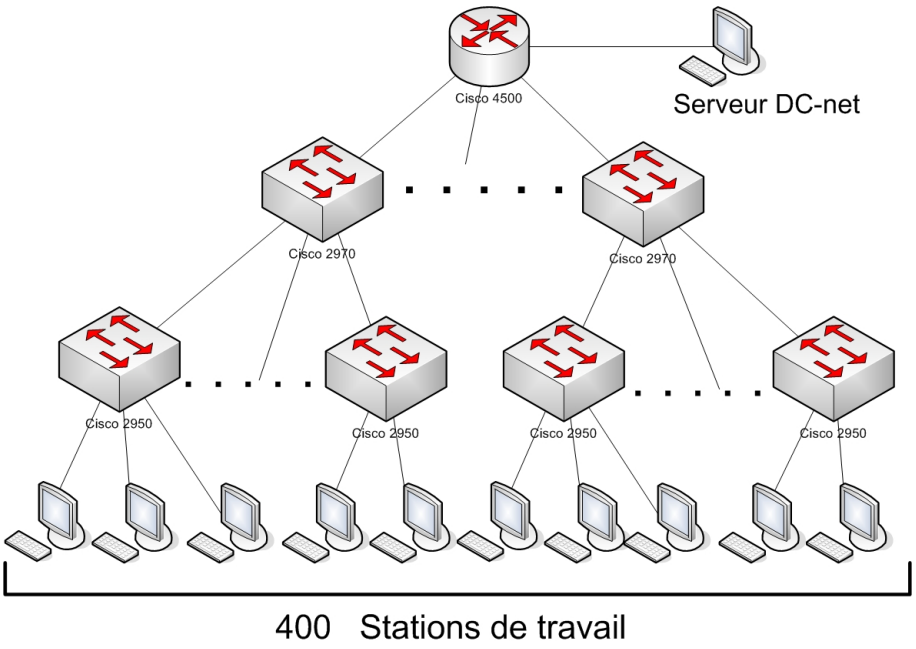


FIG. 2.5: Le réseau de test.

et les résultats pour un petit nombre de clients autant en bande passante qu'en latence. La différence dans le cas de la bande passante apparaît quand les serveurs exécutant les clients ne sont pas capables de saturer la puissance de traitement du serveur exécutant le serveur DC-net. La différence en latence vient du fait que nous présentons les latences maximales, et non les valeurs moyennes, et pour les petites latences, la variance due à la complexité du système a un impact relativement plus important, et donne donc des maxima légèrement supérieurs.

2.2.3.2 Le réseau

Le deuxième ensemble de tests a été réalisé dans un réseau avec plusieurs centaines d'ordinateurs (voir figure 2.5). Le serveur DC-net est exécuté au sein d'une station Sun Blade 150, et les clients sur des stations Sun Blade 100 et 150. Pour des raisons pratiques, nous avons utilisé moins d'une centaine de ces stations pour nos tests. La plupart sont connectées aux switches c2950 représentés dans le diagramme, et groupées sous deux switches c2970. La diffusion est faite dans le ré-

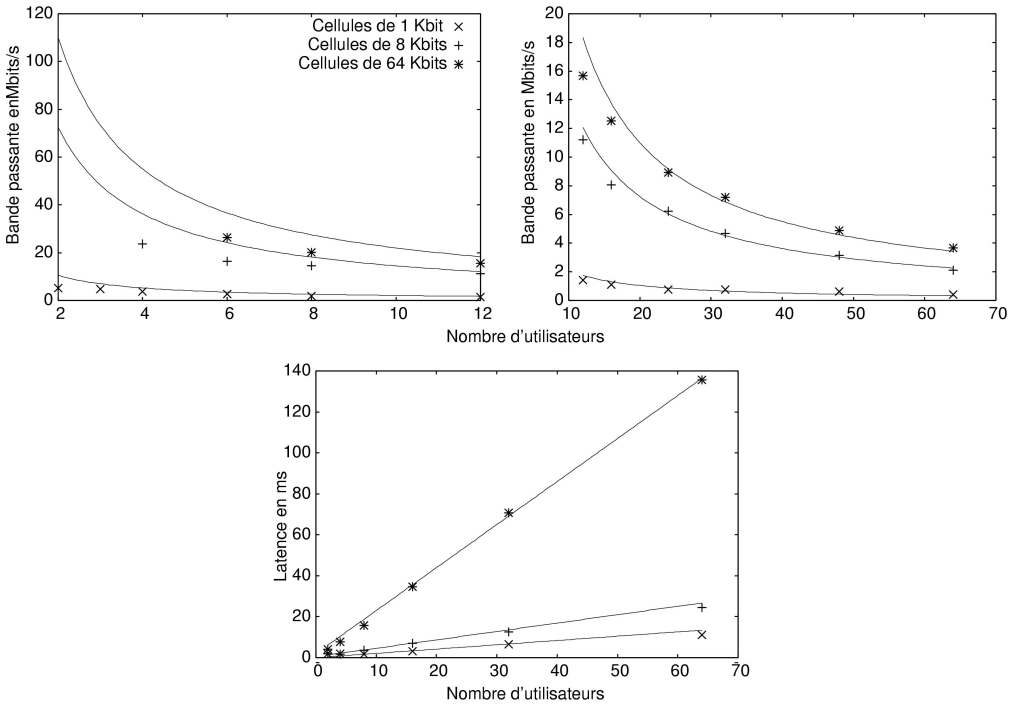


FIG. 2.6: Performances dans le réseau.

seau tout entier. Cette opération est beaucoup plus complexe dans ce contexte que dans la grappe. Comme les résultats de la figure 2.6 le montrent, la complexité hiérarchique et la taille du réseau ont un impact important autant sur la bande passante que sur la latence. Pour éviter la saturation du réseau, nous n'avons pas réalisé les tests pour des cellules de soixante-quatre Kbits et moins de huit clients, ni pour des cellules de huit Kbits et moins de quatre clients.

Comparés aux résultats de la grappe, la bande passante est globalement divisée par un facteur deux. Quand le nombre de clients est petit, ce facteur est plus important. En effet, il est difficile d'obtenir une bande passante importante sans un réseau entièrement dédié à l'envoi superposé, puisque des événements asynchrones viennent souvent briser la régularité des tours du serveur DC-net.

L'évolution de la latence est plus complexe. Pour des cellules de huit Kbits, elle est doublée par rapport aux tests dans la grappe, ce qui est en accord avec l'évolution de la bande passante. Cependant, autant pour un Kbit que pour soixante-quatre Kbits, elle est pratiquement triplée. La latence moyenne est en fait doublée quelle

que soit la taille des cellules mais l'écart-type augmente de façon importante pour les cellules de taille un et soixante-quatre Kbits, ce qui a un impact important sur les maxima. Le switch c2950 semble être le facteur limitant pour les cellules de un Kbit. Ceci n'arrive pas dans le cas de la grappe puisque le switch ne reçoit pas d'autre trafic que celui des clients du serveur DC-net alors que les c2950 reçoivent des rafales de messages de façon asynchrone. Pour les cellules de taille soixante-quatre Kbits, il n'y a pas d'explication simple. Nous pensons que la latence est accrue par des limitations de diffusion dans le réseau mais nous avons été incapables de le prouver.

D'après ces résultats expérimentaux, un serveur DC-net nous permet d'implémenter dans un réseau local un service de voix sur IP avec des groupes anonymes de centaines d'utilisateurs. Dans le cas de la vidéo-conférence sur IP, ces groupes peuvent atteindre cent ou deux cents utilisateurs, mais le nombre de communications simultanées doit rester restreint. Ainsi, s'il y a cent utilisateurs, il ne pourra pas y avoir plus de huit communications actives ; s'il y a deux cent utilisateurs, le nombre maximum se voit réduit à quatre. En effet la bande passante que le serveur DC-net met à la disposition des utilisateurs doit être divisée entre ceux qui l'utilisent en même temps. Peu d'applications pourront être ainsi implémentées en cas de concurrence avec plus de quelques dizaines d'ordinateurs.

2.2.3.3 Mesures sur l'Internet

Nous avons pas réalisé des mesures sur l'Internet pour plusieurs raisons. D'abord, par manque de temps et d'une infrastructure appropriée. En effet, pour que les résultats des tests soient représentatifs des limitations inhérentes au serveur et non pas à une connexion Internet donnée, il faudrait réaliser des tests sur un grand nombre de lignes ADSL avec des fournisseurs variés et tester plusieurs distributions des clients sur différents points du monde. Tester ceci entre des laboratoires de recherche aurait été possible, mais peu représentatif en raison des connexions privilégiées dont bénéficient ces laboratoires.

Deuxièmement, pour qu'un tour d'envoi superposé soit exécuté, il faut que tous les participants émettent un message brouillé. Il suffit que l'un d'eux ne le fasse pas pour que le résultat du tour ne soit pas calculable. De plus, comme nous l'avons indiqué dans la section 2.2.1, les émissions des clients se font à travers des tampons, et en cas de déconnexion d'un utilisateur il est généralement nécessaire de revenir en arrière sur plusieurs tours. Si de nos jours on obtient de bonnes performances

dans les connexions ADSL ou UMTS, il ne faut pas oublier que celles-ci sont très variables d'une connexion à l'autre, et d'un instant à l'autre sur une même connexion. En utilisant un protocole d'envoi superposé, la latence pour un tour est toujours supérieure à la latence la plus grande de tous les utilisateurs, et globalement le service fonctionne comme si tous les utilisateurs avaient une connexion qui a toujours la plus mauvaise des latences de toutes les connexions.

Finalement, les limitations que nous nous sommes fixées en occupation de bande passante, notamment en émission, pour les connexions de type ADSL ou UMTS font que cette étude, quoi qu'intéressante, n'est pas indispensable. En effet, la vidéo-conférence sur IP n'est pas possible, tant avec une connexion de type ADSL qu'avec une connexion de type UMTS, et le nombre maximum de communications simultanées est limitée à une dans le cas de la voix sur IP (pour l'UMTS on dépasse d'ailleurs légèrement la limite fixée). Si les performances se dégradent de la même façon qu'en passant d'une grappe à un réseau réel (ce qui nous donne juste une idée de l'ordre de grandeur), on devrait pouvoir avoir des groupes anonymes de l'ordre de la centaine. Cependant, il semble peu probable qu'avec des groupes d'au delà de quelques dizaines d'utilisateurs, une seule communication puisse suffire aux besoins.

2.2.3.4 Utilisation de mandataires

Enfin, nous avons réalisé des tests sur l'impact de l'utilisation des mandataires. Nous avons choisi soixante-quatre ordinateurs et nous les avons organisés en quatre topologies virtuelles : sans mandataires, et avec un, deux et trois niveaux de mandataires. La figure 2.7 montre la topologie avec deux niveaux de mandataires.

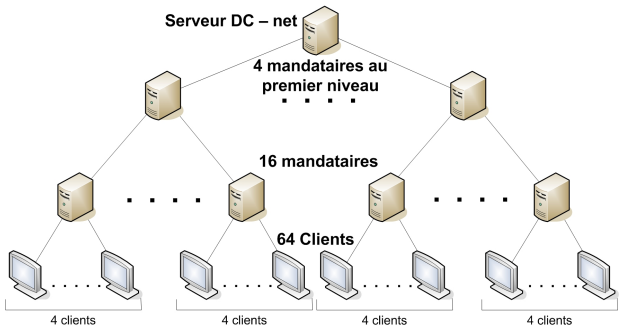


FIG. 2.7: Utilisation de mandataires.

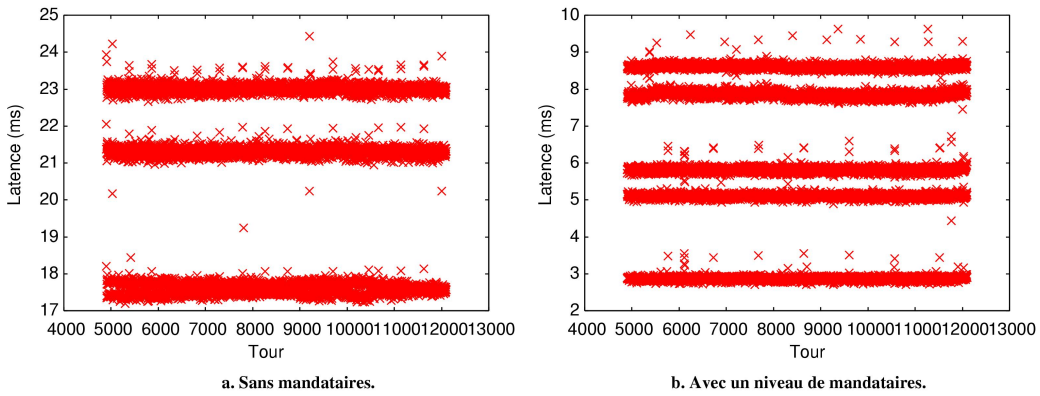
Les résultats des tests pour des cellules de taille huit Kbits sont comparés aux valeurs espérées. Par exemple, l'utilisation de huit mandataires avec huit clients chacun, divise par huit le coût de traitement du serveur par rapport à un serveur avec soixante-quatre clients, mais chaque tour est réalisé en deux phases. Étant donné que le nombre de clients est divisé par huit, la bande passante maximale que chacun des mandataires et le serveur DC-net devraient être capable de gérer se voit multipliée par huit. Chaque tour étant réalisé en deux phases, la taille de tampon des clients doit en général être doublée et la latence ne sera en principe que quatre fois mois importante qu'avec un seul serveur. Dans la figure 2.8, les valeurs mesurées expérimentalement sont comparées aux valeurs espérées avec mandataires, calculées à partir des mesures sans mandataires (première colonne).

Niveaux de mandataires	0	1	2	3
Bande passante espérée (Mbits/s)	—-	17.0	33.9	67.8
Bande passante mesurée (Mbits/s)	2.12	10.1	12.2	11.1
Latence espérée (ms)	—-	6.11	4.58	3.05
Latence mesurée (ms)	24.4	9.62	8.47	9.41

FIG. 2.8: Impact des mandataires sur la performance.

Dans la réalité, les mandataires pourraient être eux-mêmes des clients du serveur DC-net. Afin que les résultats soient plus simples à interpréter, nous avons décidé que les mandataires agissent en tant qu'ordinateurs dédiés et non pas en tant que clients. Les phénomènes observés ci-dessous sont similaires dans les deux cas.

La table 2.8 montre que les résultats sont beaucoup moins bons qu'espéré. Nous avons observé (voir figure 2.9) que généralement le comportement du système est proche de ce qui est espéré, mais le nombre de tours très lents et la durée maximale de ceux-ci augmente avec l'introduction de mandataires, réduisant du même coup la performance globale.



a. Sans mandataires.

b. Avec un niveau de mandataires.

FIG. 2.9: Mandataires et latence.

Nous pensons que ceci vient du fait que dans un tour d'envoi superposé, le serveur doit attendre que le dernier des clients complète le tour. Ainsi, si deux ordinateurs sont lents, seulement le plus lent des deux impose sa latence au tour. S'ils sont placés dans différents niveaux hiérarchiques ces deux ordinateurs vont ajouter leur latence en causant une dégradation des performances. Enfin, comme les résultats pour deux et trois niveaux l'indiquent, si les groupes connectés aux mandataires sont trop petits, les changements de contexte, et l'émission et réception de paquets seront les facteurs limitants et il n'y a pas de gain à distribuer les calculs.

2.3 Conclusions

Les serveurs DC-net permettent d'obtenir des groupes anonymes plus importants que les EBBS dans les réseaux locaux. Cependant, nous avons pu observer lors des tests réalisés que le débit obtenu par les émissions par envoi superposé chutait lorsque le nombre ou la disposition des clients utilisés pour les tests changeait. Ainsi, il a fallu à chaque test paramétrer manuellement le serveur, et réaliser des

implémentations spécifiques pour chaque système d'exploitation. Obtenir un serveur qui puisse être performant pour des contextes d'utilisation variés semble un problème difficile. Ceci, avec la faible portabilité des implémentations (notamment pour l'intégration au sein de la pile IP), est sans doute l'une des raisons expliquant l'absence d'implémentation opérationnelle de serveurs DC-net de nos jours.

Lorsque les utilisateurs sont distribués sur l'Internet, l'EBBS semble être une meilleure option que le serveur DC-net et ce pour plusieurs raisons. Premièrement, les limitations en émission qui de nos jours caractérisent les connexions en ADSL ou en UMTS, font que les serveurs DC-net soient mal adaptés à ce contexte et se voient restreints à des applications où il y a peu ou pas de communications simultanées. Étant donné que le nombre de communications actives augmente en général avec le nombre d'utilisateurs inscrits au serveur, ceci limite aussi fortement la taille des groupes anonymes. Deuxièmement, l'émission par envoi superposé nécessite la collaboration de tous les utilisateurs connectés au serveur. L'importance de la variance en débit et latence des lignes dont peuvent disposer les utilisateurs sur Internet de nos jours fait que les performances d'un tel système chutent drastiquement dès qu'une poignée d'utilisateurs sont connectés à un tel serveur.

Le serveur DC-net semble donc être la meilleure approche pour obtenir des grands groupes anonymes dans des réseaux locaux. L'EBBS permet de gérer des groupes de taille moyenne dans les réseaux locaux mais il est surtout beaucoup plus simple à implémenter et paramétrer. Sur l'Internet la seule approche viable semble être l'utilisation d'un EBBS et pour la plupart des applications les groupes anonymes seront limités à une très petite taille.

Dans le chapitre suivant nous présentons une primitive qui peut apporter d'autres solutions à notre problème que les EBBS et serveurs DC-net. Ces nouvelles solutions, et leurs variantes seront introduites et analysées dans le chapitre 4.

Chapitre 3

Le PIR

En général, pour récupérer un élément d'une base de données, un utilisateur envoie une requête indiquant quel élément l'intéresse, puis la base lui renvoie l'élément en question.

Quel élément de la base de données intéresse un utilisateur peut être une information que celui-ci souhaite garder en secret, même auprès des administrateurs de la base. Par exemple la base peut être :

– une bibliothèque électronique, et quels livres nous lisons peut fournir des informations sur nos convictions politiques, ou certains détails de notre personnalité qu'il peut être souhaitable de garder confidentiels ;
– une base de données pharmaceutique, et certains laboratoires clients de la base peuvent désirer qu'on ne puisse pas savoir à quels principes actifs ils s'intéressent ;
– des cours d'actions, et les clients peuvent être des investisseurs ne voulant pas dévoiler quels cours les intéressent.

Une manière évidente de résoudre ce problème consiste pour un utilisateur à télécharger toute la base, et à extraire localement l'information qui l'intéresse. Ceci est en général inacceptable, voir même impossible si la base est de taille extrêmement importante (par exemple une bibliothèque électronique), confidentielle (par exemple, une base de données pharmaceutique), ou rapidement obsolète (par exemple, des cours d'actions).

3.1 Notions de base

Un protocole PIR[1] est un protocole permettant à un utilisateur d'obtenir un élément d'une base de données, sans dévoiler aucune information, même aux gestionnaires de la base, sur quel élément particulier l'intéresse, avec un transfert de données sous-linéaire par rapport à la taille de la base.

Dans la littérature, un protocole PIR est défini comme un protocole permettant de récupérer un seul bit d'une base de données. La base est alors représentée comme une chaîne de n bits dont un utilisateur veut récupérer le i-ème bit sans dévoiler i aux administrateurs de la base. Les protocoles permettant de récupérer plus d'un bit sont dits des protocoles de récupération de bloc privée (*Private Block Retrieval* ou PBR). Ce bloc est usuellement considéré comme un ensemble de k bits contigus de la chaîne de n bits.

Dans la suite, nous préférons représenter la base de données comme un ensemble de n éléments de l bits (avec $l = 1$ si les éléments ne font qu'un bit). Ceci donne une description plus intuitive de la base et nous permet de bien faire la différence entre les paramètres de la base (nombre d'éléments et taille de ceux-ci), et les spécifications du protocole (taille des blocs qui peuvent être encodés dans une réponse de la base). En effet, le nombre de bits par réponse de la base de données, c'est-à-dire la *taille des blocs*, que permet de récupérer un protocole est propre à celui-ci et indépendant de la base de données pour laquelle on l'utilise. Ces deux notions sont souvent mélangées dans la littérature et compliquent l'analyse des performances des protocoles.

Il est très simple d'adapter un protocole PIR à toute valeur de l. Par exemple, si un certain protocole PIR permet de récupérer des blocs de un bit, il peut être utilisé pour récupérer un élément d'une base de données où chaque élément a une taille de deux bits. Quand l'utilisateur envoie une requête destinée à récupérer le i-ème élément de la base, celle-ci va agir de la façon suivante :

– elle génère une réponse pour la requête de l'utilisateur en l'appliquant sur la chaîne de n bits formée par le premier bit de chacun des éléments contenus dans la base,
– elle génère une deuxième réponse en appliquant la même requête sur la chaîne de n bits formée par le deuxième bit de chacun des éléments contenus dans la

[1] Rappel : PIR (de l'anglais *Private Information Retrieval*) est un acronyme pour la récupération d'informations privée.

base.

Ce raisonnement est bien sûr généralisable pour des éléments de taille quelconque et pour des protocoles permettant de récupérer des blocs de n'importe quelle taille. Ce mode opératoire est possible car les requêtes générées dans les protocoles PIR sont toujours, à notre connaissance, indépendantes du contenu de la base. On dira que la base répond itérativement jusqu'à envoyer l'ensemble de l'élément de l bits.

3.2 PIR parfaitement sûr

Les premiers protocoles PIR furent présentés par Chor, Goldreich, Kushilevitz et Sudan en 1995 [CGKS95]. Dans cet article, les auteurs proposent d'implémenter des protocoles PIR sur des bases de données répliquées. Ces protocoles sont sûrs du point de vue de la théorie de l'information tant que les répliques de la base ne forment pas des collusions contre les utilisateurs.

Nous expliquons ici l'idée de base pour un protocole PIR permettant de récupérer des blocs de un bit. Le lecteur peut se référer aux articles [Amb97, BIKR02] pour plus de détails et des descriptions de protocoles plus performants. Nous avons inclus cette section par souci de complétude : le contenu est à la fois simple et important pour avoir une bonne vision globale, mais n'est pas indispensable à la compréhension du reste du mémoire. Le lecteur est libre de le lire par curiosité personnelle où de passer directement à la section suivante.

Exemple 3.1 Protocole PIR sur une base de données répliquée.

Soit une base de données de n éléments de un bit chacun, que nous notons $\{x_1, ..., x_n\}$, répliquée sur deux sites BDD_1 et BDD_2. Pour récupérer le i-ème bit x_i de la base de données, un utilisateur U suit le protocole suivant :

1. U génère une chaîne aléatoire de n bits notée $S = \{s_1, ..., s_n\}$ et l'envoie à BDD_1,
2. U inverse le i-ème bit de S, et envoie le résultat $S' = \{s'_1, ..., s'_n\}$ à BDD_2.

Quand BDD_1 reçoit S, elle réalise le XOR de tous les x_j tels que $s_j = 1$ et renvoie le résultat R_1 à U. BDD_2 réalise le même calcul en utilisant S' et renvoie le résultat R_2 à U.
Après réception de R_1 et de R_2, U calcule $x_i = R_1 \oplus R_2$.

Si la base de données contient des éléments ayant l bits, celle-ci opère itérativement sur chacun des bits, comme indiqué dans la section précédente (le protocole décrit dans cet exemple est particulièrement intéressant si $l > n$).

Dans l'exemple donné ci-dessus, le seul bit de la base de données dont on réalise le XOR une seule fois est x_i, puisque pour tout $j \neq i$, on a $s_j = s'_j$, et donc x_j est "inclus" dans R_1 et dans R_2. BDD_1 ne reçoit de la part des utilisateurs que des requêtes parfaitement aléatoires, donc ne comportant aucune information, en particulier sur l'élément qui intéresse l'émetteur d'une requête. De même, BDD_2 ne reçoit que des requêtes dont tous les bits ont une chance sur deux d'être à un, c'est-à-dire, des chaînes de bits parfaitement aléatoires. L'information sur l'élément intéressant l'utilisateurs se trouve dans la différence entre la requête envoyée à BDD_1 et la requête envoyée à BDD_2. On peut donc être parfaitement sûr que si les deux sites ne forment pas une collusion contre un utilisateur et ne comparent pas les requêtes qu'ils reçoivent, ils ne peuvent tirer aucune information de celles-ci.

Les articles citées précédemment dans cette section présentent, des protocoles pour des bases de données distribuées sur plus de deux sites. Il est possible d'obtenir des protocoles où il faut que tous les sites forment une collusion contre un utilisateur afin d'obtenir des informations sur l'élément l'intéressant. Cependant, avec des tels protocoles, les utilisateurs doivent espérer que certains sites ne vont pas former des collusions contre eux. Pour cela, on peut faire en sorte que les sites soient gérés par des administrateurs différents, et choisir ces administrateurs de telle sorte qu'une collusion soit peu vraisemblable. Ceci est en général une contrainte importante, qui peut être rédhibitoire dans certains cas, en particulier si le contenu de la base de données est confidentiel.

Une deuxième contrainte importante est que pour que l'utilisateur obtienne le résultat souhaité, les répliques de la base de données doivent être dans un état cohérent tout au long de l'obtention des réponses des sites. En effet, si pour une requête un site réalise un calcul avec un contenu $\{x_1, ..., x_n\}$ donné, et un autre site réalise le calcul avec un autre contenu $\{x'_1, ..., x'_n\}$ le résultat du protocole sera indéterminé. Maintenir cette cohérence est coûteux, tout particulièrement pour les bases de données évoluant rapidement.

3.3 PIR sur des bases de données non-répliquées

Pour les raisons qui viennent d'être indiquées, nous ne nous intéressons qu'aux les protocoles PIR opérant sur des bases de données non-répliquées. La sécurité de tels protocoles ne peut être assurée contre des attaquants ayant une puissance de calcul illimitée. En effet, il est prouvé [CGKS95], qu'il n'existe pas de protocole PIR ayant une communication sous-linéaire et ne reposant pas sur la réplication, qui soit sûr du point de vue de la théorie de l'information. Le premier des protocoles PIR opérant sur des bases de données non-répliquées fut présenté par Kushilevitz et Ostrovsky en 1997 [KO97], et depuis plusieurs autres solutions ont été proposées [Ste98, Man04, CMS99, Cha04, Lip05, GR05].

Tous ces protocoles suivent une approche similaire, mais il est parfois difficile de distinguer quelles sont les différences fondamentales entre ceux-ci et les innovations que chacun d'entre eux apporte. La section suivante présente l'approche suivie par tous ces protocoles en indiquant en quoi chacun de ces articles est un progrès dans la recherche des protocoles PIR sans réplication [Agu06a].

3.4 Évolution des protocoles PIR

Dans [KO97], Kushilevitz et Ostrovsky présentent le premier protocole PIR ne nécessitant pas que la base de données soit répliquée, en utilisant des *résidus quadratiques*. La définition exacte de ce qu'est qu'un résidu quadratique est sans importance dans le contexte de cette section et on ne s'intéressera qu'à certaines des propriétés fondamentales de ces nombres :

- l'utilisateur peut de façon efficace générer des nombres qui sont des résidus quadratiques (RQ), et des nombres qui sont des résidus non-quadratiques (RNQ),
- l'utilisateur peut de façon efficace tester si un nombre est un RQ ou un RNQ,
- l'utilisateur peut envoyer des ensembles de tels résidus à une base de données qui sera incapable de distinguer les RQs des $RNQs$,
- il y a une opération OP, calculable par une telle base, qui d'un ensemble de RQs et de $RNQs$ donne un résidu qui est un RNQ si et seulement si le nombre de $RNQs$ en entrée de l'opération est impair.

Ce protocole permet de récupérer un seul bit. On considérera pour simplifier que la base est donc composée d'éléments de un bit. Si les éléments de la base sont

composés de plus d'un bit, celle-ci peut procéder itérativement pour l'envoi des éléments comme décrit dans la section précédente. L'idée principale de ce protocole est d'envoyer un résidu par bit dans la chaîne de n bits composant la base, le résidu associé au i-ème bit étant le seul RNQ. La base réalise l'opération OP sur l'ensemble des résidus associés aux bits de la base dont la valeur est de un (et ignore les autres résidus), puis envoie le résidu résultant de l'opération à l'utilisateur. Si le i-ème bit a pour valeur un, l'entrée de OP contient un RNQ et le résidu reçu par l'utilisateur est donc aussi un RNQ. Si le i-ème bit est à zéro, la base ne sélectionne que des RQs, et le résidu reçu par l'utilisateur est un RQ. La figure 3.1 résume ceci.

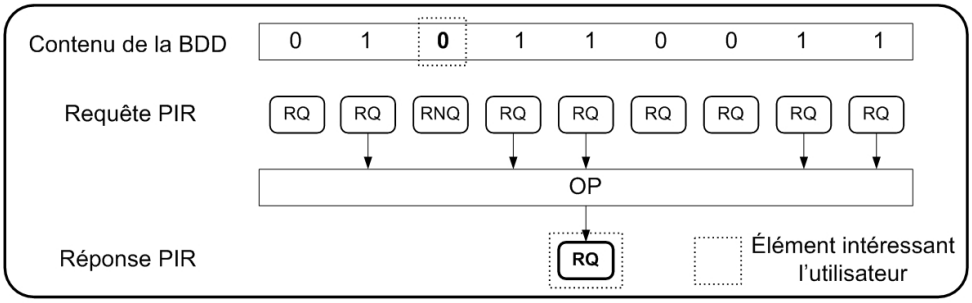


FIG. 3.1: Résidus et protocoles PIR.

Avec un tel protocole, l'utilisateur doit envoyer n résidus et la base de données en renvoie un seul. Le protocole proposé par les auteurs inclut une technique pour équilibrer et réduire fortement le coût des communications, technique qu'ils reprennent de l'article fondateur des protocoles PIR [CGKS95]. L'idée est de ne pas représenter la base de données comme une chaîne de n bits mais comme une matrice de s lignes et t colonnes (avec $s \times t = n$). L'utilisateur envoie t résidus, un par colonne dans la matrice, et la base renvoie s résidus, un pour chaque ligne dans la matrice. Le principe est analogue à celui de l'itération introduit dans la section précédente. Comme la figure 3.2 le montre, l'utilisateur récupère une colonne complète des bits de la matrice, qui contient le bit qui l'intéresse.

Le premier apport de l'article de Kushilevitz et Ostrovsky est donc un procédé pour réaliser des PIR sans réplication. Dans sa thèse de Mastaire [Man04], Eran Mann présente une formalisation de ce procédé, et introduit la notion de famille de prédicats "à trappe" homomorphiques qui est tout simplement la généralisation des prédicats ayant les mêmes propriétés que celles qu'on a décrit en début de

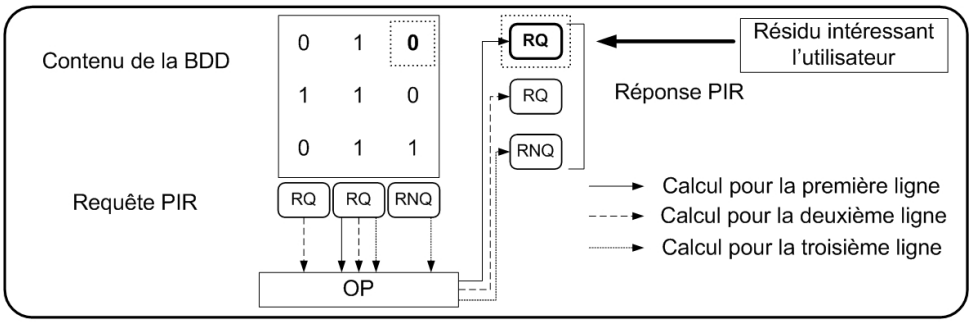


FIG. 3.2: Protocole PIR basique de Kushilevitz and Ostrovsky.

cette section pour les résidus quadratiques. Tous les protocoles PIR opérant sur des bases de données non-répliquées suivent le procédé de Kushilevitz et Ostrovsky. Ces protocoles ont utilisé diverses familles de tels prédicats ayant des propriétés qui ont permis de beaucoup améliorer les performances des protocoles PIR, mais dans chacun de ces protocoles, c'est la construction de base de ces auteurs qui est reprise.

La technique pour équilibrer les communications n'est pas une idée originale de Kushilevitz et Ostrovsky. Cependant, les auteurs ont remarqué que lorsqu'on se sert de cette technique il est possible d'utiliser récursivement le protocole PIR. L'utilisation récursive réduit la taille des requêtes PIR en augmentant la taille des réponses de la base de données et permet de rendre les protocoles PIR beaucoup plus polyvalents. C'est le deuxième grand apport de l'article de Kushilevitz et Ostrovsky.

Cette approche est possible car un utilisateur n'a pas besoin de récupérer toutes les données renvoyées par le serveur quand la technique d'équilibrage des communications est utilisée, mais uniquement le résidu associé au bit qui l'intéresse. Ainsi, au lieu de renvoyer la réponse PIR issue de la première récursion du protocole PIR, la base de données utilise cette réponse PIR comme base de donnée virtuelle pour une deuxième requête PIR envoyée par l'utilisateur.

Dans la figure 3.3, nous reprenons l'exemple donné par la figure 3.2 : la première récursion donne lieu à trois résidus qui servent comme base de données pour la deuxième récursion. On remarquera que lors de la deuxième récursion l'utilisateur a besoin de récupérer en entier le résidu qui contient l'information sur le bit l'intéressant. En reprenant l'exemple de la figure 3.2, la sortie de la première ré-

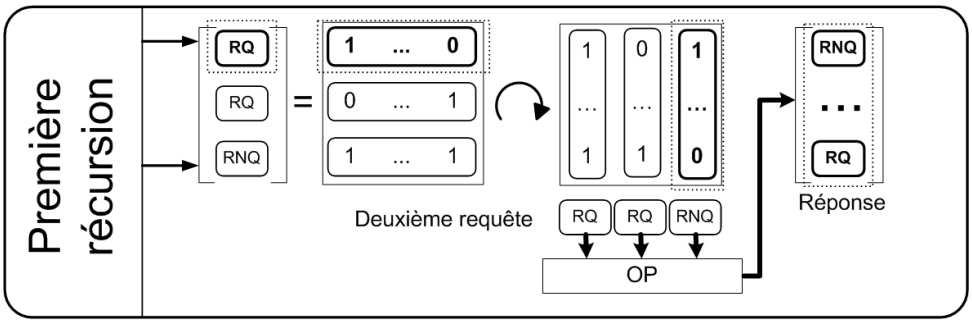


FIG. 3.3: Protocole PIR récursif de Kushilevitz and Ostrovsky.

cursion donne lieu à trois résidus, et lors de la deuxième récursion, l'utilisateur doit récupérer un de ces résidus en entier. Chaque réponse PIR générée lors de la deuxième récursion permet de coder un bit de ce résidu, et il faut donc que la base de données réponde itérativement pour pouvoir envoyer tous les bits dont l'utilisateur a besoin pour retrouver ce résidu. Il y a donc bien une augmentation de la taille de la réponse de la base et une diminution de la taille des requêtes envoyées par l'utilisateur par rapport à celles du protocole présenté dans la figure 3.1. Cette alternative est plus intéressante que la simple utilisation de l'équilibrage des communications pour deux raisons. La première n'était pas évidente lorsque l'article a été présenté et vient du fait que les protocoles PIR proposés dans les années qui ont suivi l'article de Kushilevitz et Ostrovsky sont très efficaces dans l'implémentation de la récursivité, comme on le verra plus loin. La deuxième est que l'équilibrage des communications peut se faire une seule fois si on représente la base de données comme une matrice de bits, alors que la récursion peut se faire autant de fois que de dimensions dans la représentation de la base.

En effet, dans le cas qu'on vient de voir, après la deuxième récursion, l'utilisateur a besoin de tous les résidus qui sont dans la réponse PIR et il faut lui renvoyer tous ces nombres. Cependant, si la base de données est représentée comme un parallélépipède de bits de côtés s, t, et u (avec $s \times t \times u = n$), après la première récursion calculée avec une requête de u résidus, la réponse sera un rectangle de $s \times t$ résidus dont un seul intéressera l'utilisateur. De même, après la deuxième récursion avec une requête de t résidus la réponse sera une colonne de s groupes de résidus, un seul de ces groupes intéressant l'utilisateur, et on pourra faire une troisième récursion avec cette fois-ci une requête de s résidus pour obtenir un unique grand groupe de résidus. De façon générale si la base de données est représentée par

un hyper-rectangle de dimension L, L niveaux de récursion sont possibles. Avec L niveaux de récursions, l'utilisateur doit envoyer L requêtes mais si les côtés de l'hyper-rectangle sont égaux, chacune peut être composée de seulement $n^{1/L}$ résidus. Ceci réduit nettement la taille des requêtes pour des bases avec beaucoup d'éléments, mais la taille de la réponse PIR augmente exponentiellement en L. Un compromis doit donc être choisi, en fonction de l'application pour laquelle le protocole PIR est utilisé.

En plus de la thèse de Eran Mann, l'article de Kushilevitz et Ostrovsky a donné lieu à un autre travail sur les PIR en 1998. Il s'agit d'un article par Julien Stern [Ste98], qui a permis aux protocoles PIR de faire un grand pas en avant. Stern, propose exactement le même protocole que Kushilevitz and Ostrovsky, en remplaçant les prédicats à trappe qui ne peuvent encoder qu'un seul bit d'information (par exemple, être un RQ ou un RNQ), par des systèmes de chiffrement homomorphiques. Ces systèmes de chiffrement respectent les propriétés des prédicats à trappe homomorphiques, et en plus peuvent encoder un grand nombre de bits dans la sortie de l'opération OP. Quand les utilisateurs essayent de récupérer des éléments d'une base de données de l bits avec $l > 1$, ceci est sans aucun doute très intéressant puisque chaque résultat de l'opération OP (qui d'ailleurs est d'une taille équivalente à celle des résidus utilisés précédemment), permet d'encoder un grand nombre des bits de l'élément à récupérer. Cependant, même si l'utilisateur n'est pas intéressé à recevoir un bloc d'information (mais seulement un bit), la possibilité d'encoder un grand nombre de bits dans chaque réponse est très intéressante. La raison en est assez simple : quand on utilise l'équilibrage des communications ou la récursion de Kushilevitz et Ostrovsky, la base de données renvoie un nombre pour chaque bit composant une colonne (pour l'équilibrage des communications), ou pour chaque bit composant les résidus (pour la récursion). En utilisant des systèmes de chiffrement homomorphique, les colonnes et les résidus peuvent être encodés très efficacement. Ceci est particulièrement important dans le cas de la récursion car le facteur multiplicatif donnant lieu à la croissance exponentielle de la taille des réponses se voit réduit grandement (de nos jours ce facteur est divisé par mille environ).

L'année suivante, Cachin, Micali et Stadler ont présenté un protocole [CMS99] basé sur un nouveau prédicat à trappe, qu'ils appellent la supposition ϕ. Ceci peut sembler un pas en arrière après l'introduction des systèmes de chiffrement par Stern, et si c'est le cas d'un point de vue des performances pratiques des protocoles, ce ne l'est ni pour les performances asymptotiques ni sur le plan conceptuel.

La raison principale est que ces prédicats ont une propriété très intéressante : un utilisateur peut créer un générateur compact à partir duquel la base de données peut obtenir les nombres qui constituent la requête PIR. Ceci réduit spectaculairement la taille des requêtes, et même si le système n'est pas intéressant tant que la base contient un nombre d'éléments raisonnable, quand ce nombre grandit asymptotiquement, cette approche l'emporte sur toutes les propositions précédentes. L'idée de base est de créer d'abord le générateur et *ensuite* de définir le prédicat à trappe tel que le nombre généré et associé au bit intéressant l'utilisateur soit le seul à être particulier au niveau du prédicat. Cette approche a récemment été réutilisée pour un protocole PIR très intéressant décrit à la fin de cette section.

En 2004 il y a eu une redécouverte de la proposition de Stern [Cha04], et une proposition par Lipmaa [Lip05] qui est une adaptation de la construction de Stern pour certains systèmes de chiffrement homomorphiques. Dans cet article, Lipmaa utilise les propriétés du système de chiffrement homomorphique récemment découvert par Damgärd et Jurik [DJ03] pour construire des protocoles PIR qui sont à la fois pratiques et asymptotiquement intéressants. L'article de Lipmaa en soi n'apporte pas d'innovation, mais remarque que l'utilisation correcte de ce système de chiffrement permet d'avoir une croissance uniquement linéaire dans la taille des réponses de la base de données quand on utilise la construction récursive de Kushilevitz et Ostrovsky. Ceci améliore grandement la polyvalence de ce PIR et lui donne un intérêt théorique, car son comportement asymptotique est du coup bien meilleur que celui de ses prédécesseurs[2].

Enfin, l'année dernière, Gentry et Ramzan ont présenté un protocole [GR05], qui est à la fois pratique et asymptotiquement plus performant que celui de Lipmaa, même si pour un grand nombre d'applications le protocole de Lipmaa reste le plus intéressant, comme on le verra dans la section suivante. Dans leur article, les auteurs proposent une construction qui généralise le protocole de Cachin *et al.*, et présentent une implémentation qui utilise une variante proche de la supposition ϕ. En plus de la généralisation, les auteurs introduisent des variations mineures dans le protocole. La première modification est la même que celle introduite par Stern par rapport au protocole de Kushilevitz et Ostrovsky : ils ont modifié le prédicat

[2] L'annexe A présente une analyse formelle des PIR basés sur les chiffrements homomorphiques. Ceux-ci sont le meilleur choix pour l'application que nous donnons au PIR dans le chapitre suivant et nous avons voulu dans cette annexe donner un cadre formel et un ensemble de preuves qui sont difficiles à trouver dans la littérature. Les démonstrations que nous y présentons visent en particulier à prouver la validité de la construction de Stern. Ces démonstrations sont facilement adaptables pour la construction de Lipmaa.

pour encoder plusieurs bits. La seconde modification est très simple et consiste à utiliser les même nombres (qui sont associés aux bits de la base de données) pour toutes les requêtes. Ceci est pratiquement trivial mais ne fut pas proposé par Cachin *et al.*, et permet de faire de très petites requêtes qui ne font que décrire la structure dans laquelle uniquement le nombre souhaité aura des propriétés spéciales. Cependant, cette diminution du coût est applicable à toutes les requêtes sauf la première, puisque les nombres doivent être échangés au moins une fois. Si l'utilisateur envoie une seule requête (ou un petit nombre de requêtes) à un serveur donné, le coût des communications par requête devient vite extrêmement grand.

3.5 Comparaison des performances

Le PIR ayant été défini initialement comme un protocole servant à récupérer un seul bit, on continue de nos jours encore à mélanger des notions propres à ce contexte et des notions propres au domaine d'application le plus commun pour les PIR, c'est-à-dire la récupération d'éléments ayant plusieurs bits. Ainsi, même si pour récupérer un bloc de données de taille variable en fonction de l'application il serait utile de connaître, par exemple, la taille des requêtes, et le facteur d'expansion (le nombre de bits renvoyés sur le nombre de bits de l'élément à récupérer) de la réponse de la base de données, la plupart des articles ne présentent les performances des PIR que par le coût global de la transaction. Certains auteurs, indiquent comme remarque a posteriori quel est le *taux de transfert* ou l'*évolution du coût relatif quand on récupère des grands blocs*.

Une telle approche a un impact négatif sur la clarté des résultats. Ainsi, par exemple, les titres des articles [Cha04, Lip05, GR05] indiquent respectivement une performance logarithmique, polylogarithmique, et constante dans le coût des communications en fonction du nombre d'éléments dans la base de données, mais chacun d'entre eux utilise une mesure différente pour évaluer cette performance. Un autre exemple est celui de la quête pour un protocole PIR avec un coût en communication logarithmique en fonction du nombre d'éléments dans la base. En effet, pour récupérer un bit dans une base de données classique (sans protocole PIR), le client doit envoyer une requête de $log(n)$ bits (l'index) à la base, et la base doit renvoyer un seul bit. Le coût total des communications est de $(log(n)+1)$ bits, et donc savoir si c'est possible de récupérer un bit suivant un protocole PIR avec un coût logarithmique est une question évidemment intéressante. Mais, cette approche n'est

absolument pas adaptée à la récupération de blocs, puisque dans ce cas la question fondamentale est de savoir s'il est possible d'avoir un protocole PIR avec des requêtes qui aient un coût logarithmique en fonction du nombre d'éléments dans la base de données et un facteur d'expansion constant dans les réponses de la base. Cependant, même si les protocoles PIR ont plus de chances d'être utilisés pour récupérer des blocs d'information, la plupart des articles comparent le comportement asymptotique du coût global des communications à $O(log(n))$, ce qui est peu pratique pour le lecteur, mais en plus théoriquement moins intéressant que d'évaluer le rapprochement à la limite de $O(log(n))$ pour la requête et constante pour la récupération de blocs. Ainsi, par exemple Stern affirme que sa construction permet d'obtenir un coût sous-polynômial, mais obtenir de requêtes de cette taille avec sa construction implique un facteur d'expansion super-polylogarithmique. De même Lipmaa, réussit à avoir un coût de communication polylogarithmique, cependant l'évolution asymptotique est polylogarithmique non seulement pour les requêtes, mais aussi pour le facteur d'expansion.

Pour ces raisons nous proposons un ensemble de mesures qui diffèrent de celles traditionnellement utilisées. Nous avons préféré ne pas utiliser le facteur d'expansion pour la réponse de la base de données comme mesure et ceci pour deux raisons. La première est que la taille des blocs qu'un protocole PIR permet de récupérer est souvent fixe. Le facteur d'expansion dépend donc de la taille des éléments à récupérer, en particulier si les éléments sont plus petits que les blocs qui peuvent être encodés par le protocole, et ainsi varie en fonction de l'application. Nous attribuons une grande importance à séparer les mesures décrivant la base de données et celles décrivant les protocoles PIR, chose qui a souvent été négligée dans ce domaine et qui rend les discours souvent beaucoup plus compliqués que nécessaire. La deuxième raison pour laquelle nous n'utilisons pas le facteur d'expansion de la réponse c'est que les mesures, plus fondamentales, que nous proposons pour évaluer les protocoles permettent facilement de savoir pour chaque application, quel sera le facteur d'expansion de ces réponses.

Nous proposons que les protocoles PIR soient mesurés par :
- le coût de la mise en place du protocole que nous appelons le coût d'initialisation,
- la taille des requêtes,
- la taille des réponses de la base de données,
- et le nombre de bits qui peuvent être encodés par réponse, que nous appelons la taille des blocs.

Quand un utilisateur veut récupérer des éléments de taille l dans une base de données et utiliser un protocole PIR qui propose des blocs de taille supérieure à $\alpha \times l$, α étant un entier, il existe une amélioration triviale : la base peut être vue comme composée de n/α éléments de taille $\alpha \times l$. Ceci permet de réduire la taille des requêtes sans augmenter la taille des réponses fournies par la base. Comme cette amélioration est commune à tous les protocoles PIR et dépend de l'application pour laquelle on utilise un protocole PIR, on ne l'inclura pas dans les résultats, laissant le lecteur l'évaluer par lui même pour chaque application.

L'utilisation récursive du protocole proposée par Kushilevitz et Ostrovsky, est indiquée dans les résultats par un paramètre que nous avons noté L. Dans le cas où aucune récursion n'est faite, les résultats restent valables, avec $L = 1$. L'entier k représente le nombre de bits d'un entier difficile à factoriser. Ce paramètre de sécurité ne devrait pas être choisi de nos jours inférieur à 1024. Pour le protocole de Cachin *et al.* nous avons noté le paramètre de sécurité par K. La raison de cette notation différente est que les auteurs de ce travail ont imposé des contraintes fortes sur ce paramètre pour l'évaluation asymptotique du protocole, à savoir $K > log^2(n)$, alors que la sécurité semble se baser sur le fait qu'un entier de taille K^5 ne soit pas factorisable (en plus de la supposition ϕ bien sûr). Ceci impose $K^5 > log^{10}(n)$, ce qui est beaucoup plus contraignant que $K > log^3(n)$, c'est-à-dire l'estimation asymptotique usuellement faite pour les problèmes liés à la factorisation [GR05]. Enfin, s représente un paramètre entier qui doit être fixé par l'utilisateur. Le coût des calculs pour les protocoles faisant intervenir ce paramètre étant au meilleur des cas en $O((sk)^2)$, s doit être choisi proche de un, pour limiter le coût de ceux-ci.

Protocole	KO	Stern	CMS	Lipmaa	GR
Initialisation	k	$3 \times k$	0	$3 \times k$	$n^{1/L} \times k$
Requêtes	$L \times k \times n^{\frac{1}{L}}$	$L \times (s+1) \times k \times n^{\frac{1}{L}}$	$K^4 + 2 \times K^5$	$L \times \left(s + \frac{L+1}{2}\right) \times k \times \left(n^{\frac{1}{L}} - 1\right)$	$2 \times L \times k$
Réponses	k^L	$(s+1) \times \left(\frac{s+1}{s}\right)^{L-1} \times k$	K^5	$(s+L) \times k$	$k \times 5^{L-1}$
Blocs	1	$s \times k$	1	$s \times k$	$k/5$

Tableau 3.1: Comparaison des performances.

Les résultats présentés dans le Tableau 3.1 pour le protocole de Kushilevitz et Ostrovsky (que nous notons KO) ne sont pas les mêmes que ceux du protocole présenté dans l'article par les auteurs. Dans le protocole original, la base de données ne réalisait que $L - 1$ récursions pour une représentation de son contenu en L dimensions, la raison principale pour ceci étant que le coût total des communications se voit réduit quand l'utilisateur ne récupère qu'un bit. De nos jours, les

spécialistes ont bien conscience que l'utilisation du PIR pour récupérer un bit est en principe marginale et on préfère minimiser la taille des réponses de la base de données en réalisant L récursions. Nous avons présenté les résultats du protocole KO lorsque ces L niveaux de récursions sont appliqués afin de faciliter la comparaison entre les protocoles. Les protocoles de Stern et Lipmaa, quant à eux, sont implémentables avec différents systèmes de chiffrement homomorphique. Les résultats présentés dans le tableau correspondent à une implémentation avec le système de chiffrement de Damgärd et Jurik [DJ03] qui est de nos jours le système de chiffrement homomorphique le plus efficace et polyvalent. Gentry et Ramzan ne proposent pas dans leur article l'utilisation récursive de leur protocole en suivant le modèle de Kushilevitz et Ostrovsky. Celui-ci est peut-être le meilleur exemple que nous puissions donner pour illustrer pourquoi nous pensons que les principes introduits et utilisés pour améliorer les protocoles PIR depuis KO sont peu clairs et nécessitent d'une présentation plus structurée et globale. Il n'y a aucune raison pour ne pas utiliser le protocole de Gentre et Ramzan (GR) récursivement, et la polyvalence de ce protocole se voit d'ailleurs grandement accrue si on l'utilise. Enfin, pour le protocole de Cachin *et al.* (CMS), nous n'avons pas inclus L dans les résultats car c'est un protocole théorique qui n'est pas conçu pour être implémenté. La raison pour laquelle il a été inclus dans le Tableau 3.1 est le souci d'illustrer l'impact que les innovations apportées par ce protocole ont eu sur les performances.

En effet, les résultat du tableau reflètent bien l'impact des différentes innovations présentées dans la section précédente. La réduction drastique de la taille des réponses quand la récursion est utilisée entre la première et la deuxième colonne est la conséquence directe de l'introduction par Stern de protocoles PIR qui peuvent encoder dans une réponse des blocs de taille supérieure à l'unité. La taille des requêtes est indépendante du nombre d'éléments dans la base de données[3] uniquement dans la troisième et la cinquième colonne, résultant de l'idée de Cachin *et al.* de générer les prédicats à trappe après avoir défini les nombres formant les requêtes et associés aux éléments de la base. L'utilisation des propriétés spécifiques du système de chiffrement de Damgärd et Jurik par Lipmaa, réduit la croissance de la taille des réponses d'exponentielle à linéaire en fonction de L. Enfin,

[3] Pour une étude asymptotique on doit supposer $k > log^3(n)$ [GR05] et $K > log^2(n)$ [CMS99]. De plus le protocole GR présuppose qu'il y a assez de nombres premiers inférieurs à $2^{k/5}$, cependant pour toute application pratique il n'y a aucun besoin d'augmenter k au-delà de la limite déjà fixée pour la factorisation. En effet, pour $k = 1024$, les résultats ici présentés sont valides tant que $n < 10^{58}$

la possibilité d'encoder des blocs de bits dans les réponses de la base de données, et le remplacement des générateurs de nombres de Cachin *et al.* par un ensemble de nombres premiers fixes dans l'initialisation que Gentry et Ramzan proposent, donne lieu au premier protocole pour lequel (si on exclut la phase d'initialisation), le coût des communications est indépendant du nombre d'éléments dans la base, tout en étant utilisable dans la pratique.

Lors de la récupération d'éléments de taille supérieure à celle des blocs qui peuvent être encodés par le protocole PIR le facteur d'expansion sera d'environ *(Taille des réponses)/(Taille des blocs)*. Pour la récupération d'éléments plus petits, le facteur d'expansion sera variable mais le coût des communications sera exactement d'une requête pour le client et d'une réponse pour le serveur. À l'aide de ces indicateurs on peut extraire du Tableau 3.1 des informations basiques sur les différentes applications pour chacun de ces protocoles. La récupération d'éléments de grande taille doit se faire systématiquement avec le protocole de Lipmaa qui est celui qui donnera lieu au plus petit facteur d'expansion pour la réponse. En revanche, pour obtenir des petits éléments le protocole GR est en général d'une grande efficacité. Cependant, le coût d'initialisation pour ce protocole peut être trop important pour des bases de données contenant un grand nombre d'éléments, si les utilisateurs ont peu de requêtes à faire, voire même impossible à utiliser, par exemple avec un dispositif mobile. Dans ce cas, le protocole de Stern sera souvent un meilleur choix de remplacement que celui de Lipmaa car les requêtes sont beaucoup plus petites quand L est grand. Pour illustrer ceci nous donnons trois exemples :

– un vidéo-club en ligne proposant 10000 films en MPEG4 de dix Gbits chacun,
– une base de données contenant les cours dans toutes les bourses de la planète, avec cent mille éléments, chacun de deux Kbits,
– une base de données gouvernementale contenant des dossiers sur des citoyens, avec deux cents millions d'éléments, chacun de cinq Kbits.

Pour limiter les coûts de calcul, on suppose que $s = 2$ pour les protocoles de Lipmaa et de Stern. Dans le premier exemple, la taille des éléments fait que le protocole GR est trop coûteux. Le protocole de Lipmaa est le meilleur choix et en fixant $L = 1$ afin de limiter le facteur d'expansion de la réponse, les utilisateurs peuvent récupérer les films en envoyant des requêtes de 30 Mbits, et recevoir les films avec un facteur d'expansion de 1.5.

Dans le deuxième exemple le protocole GR est clairement le meilleur choix, d'autant plus qu'étant donné la nature de cette base les utilisateurs réaliseront proba-

blement de nombreuse requêtes chacun. Sans récursion, les utilisateurs doivent échanger et stocker à l'initialisation 100 Mbits de données. Cette solution pourrait être adaptée aux utilisateurs qui comptent faire de nombreuses requêtes régulièrement. Dans le reste des cas il vaut sans doute mieux utiliser le protocole avec $L = 2$, auquel cas les utilisateurs ont juste à échanger et stocker 320 Kbits de données pour l'initialisation et peuvent récupérer les éléments de la base de données en envoyant des requêtes de taille 4 Kbits et en recevant des réponses de 50 Kbits.

Le troisième exemple est un cas dans lequel le protocole GR est inutilisable sans récursion, puisque le coût d'initialisation serait de 200 Gbits. Avec $L = 2$ ce même protocole devient intéressant puisque les utilisateur ne doivent stocker que 14 Mbits de données et peuvent récupérer les éléments en envoyant des requêtes de 4 Kbits et en recevant des réponses de 125 Kbits. Il est aussi possible d'éviter l'échange et le stockage des 14 Mbits de la phase d'initialisation, en utilisant le protocole de Stern qui permet de récupérer les éléments en envoyant des requêtes de 170 Kbits et en recevant des réponses de 150 Kbits. Le protocole de Lipmaa ne peut pas fournir d'aussi bonnes performances, puisque la taille des requêtes est toujours supérieure à 512 Kbits, quelle que soit la valeur de L.

3.6 Conclusions

Les protocoles PIR permettent de récupérer un élément dans une base de données sans que ses administrateurs puissent savoir lequel. Dans ce chapitre on a présenté une mise en relief des protocoles déjà existants et on a comparé leurs performances. L'annexe A présente un ensemble de définitions formelles, et prouve la sécurité des constructions basées sur les systèmes de chiffrement homomorphique.

Certains éléments comme l'indépendance des requêtes par rapport au contenu, permettent de se dire qu'on peut faire plus que lire dans une base de données. Par exemple, on peut choisir parmi plusieurs flux d'information : tous les matins on peut recevoir un journal électronique parmi tous les journaux du pays, le serveur utilisant tous les matins la même requête pour construire une réponse avec les contenus de sa base de données actualisés dans la soirée.

Plus généralement, les différents services qu'un serveur peut fournir à un client sont souvent représentables comme différentes informations et dans certains cas ils peuvent être tous calculés de façon à générer une base de données. Dans ces

cas, grâce au PIR un utilisateur va pouvoir obtenir un service parmi plusieurs, sans que les administrateurs du serveur puissent savoir exactement quel service ils ont fourni au client.

En particulier, en utilisant du PIR les administrateurs d'un serveur peuvent ne pas savoir s'ils sont en train de vous envoyer du bourrage chiffré ou des informations qui vous intéressent. Ceci peut être utilisé pour réaliser des communications anonymes au sein d'un réseau avec des relais qui, à la différence des relais traditionnels (cf. section 1.5.2), peuvent former des groupes de réception non-observable, et ceci de façon plus efficace que les serveurs DC-net. Dans le chapitre suivant, on montre comment non seulement le PIR nous permet de générer du bourrage mais aussi de faire un relais complet qui ne peut identifier quels services il rend à quels utilisateurs et qui, par conséquent, est incapable de trahir ses clients.

Chapitre 4

Le pMIX et ses variantes

Dans ce chapitre nous présentons d'abord une nouvelle option pour former des groupes de réception non-observable, l'utilisation du PIR. Pour illustrer ceci nous décrivons une variante de l'EBBS, que nous avons nommée pMIX, où la réception par diffusion avec adressage implicite se voit remplacée par l'utilisation du PIR.

Dans un deuxième temps, nous étudions dans la section 4.2 comment l'utilisation de l'envoi superposé permet d'implémenter différentes variantes de ce système. Enfin, dans la section 4.3 nous donnons un cadre général, pour illustrer comment parmi les nombreuses options qui sont disponibles pour implémenter des systèmes fournissant un anonymat de communication, une seule est présente dans la littérature : les serveurs DC-net. Nous montrons comment ces serveurs font en fait partie d'une famille beaucoup plus importante et polyvalente, que le seul usage de l'envoi superposé ne laissait entrevoir.

4.1 Le pMIX

En 1995, Cooper et Birman ont proposé un protocole [CB95] qu'ils ont appelé service de messages privé (*Private Message Service* ou PMS). Dans ce protocole, les utilisateurs peuvent échanger des messages en les écrivant dans une base de données répliquée et en les récupérant par un protocole PIR. Ce dernier protocole assure la confidentialité des choix des utilisateurs tant que les répliques de la base ne communiquent pas entre eux.

De nos jours, les protocoles PIR opérant sur des bases de données non-distribuées nous permettent de proposer un système similaire avec des échanges de messages très rapides. L'utilisation de bourrage chiffré et de fausses requêtes PIR nous permet d'implémenter un système fournissant un anonymat de communication [AD05].

Dans la section 4.1.1 nous présentons les pMIX comme des EBBS dont on remplace la diffusion avec adressage implicite par l'utilisation d'un protocole PIR, puis en introduisant progressivement des améliorations. Ensuite, nous étudions leurs performances dans la section 4.1.2.

4.1.1 De l'EBBS au pMIX

Dans cette section nous modifions un EBBS par étapes jusqu'à arriver au pMIX. Pour simplifier les notations, dès l'introduction de la première modification nous entendrons par pMIX le serveur résultant, la signification de ce terme évoluant jusqu'à la version finale, présentée en fin de section.

Nous rappelons qu'un EBBS est un serveur où à chaque tour, chaque utilisateur écrit dans un emplacement qui lui est réservé un message chiffré ou du bourrage chiffré, selon qu'il veut transmettre des informations ou pas. L'EBBS diffuse à chaque tour l'ensemble des contenus des emplacements à tous les utilisateurs (voir figure 2.1). Un utilisateur interpellé reçoit lors d'un tour d'appel un message lui indiquant à quel emplacement s'intéresser. À partir de ce moment, et jusqu'à la fin de la communication l'utilisateur ne déchiffre que le contenu de l'emplacement en question.

Nous définissons un pMIX comme un EBBS où les emplacements sont vus comme les éléments d'une base de données, et où à chaque tour chaque client envoie une requête PIR pour récupérer l'élément qui l'intéresse (en plus d'écrire dans son emplacement). À chaque tour, au lieu de diffuser le contenu de l'ensemble des emplacements comme le fait un EBBS, le pMIX va répondre à chacune des requêtes PIR reçues dans le tour. Comme pour un EBBS, les utilisateurs réalisent à chaque tour les mêmes actions vis-à-vis des administrateurs du serveur, qu'ils soient en train de communiquer ou pas. Lorsqu'ils ne communiquent pas ils écrivent du bourrage chiffré dans leur emplacement, et ils envoient des requêtes PIR aléatoires.

4.1.1.1 Utilisation

Quand un utilisateur *A* veut établir une communication anonyme avec un utilisateur *B*, ces deux utilisateurs doivent suivre le protocole de mise en place de communication indiqué ci-dessous.

Protocole 4.1 Protocole de mise en place d'une communication

```
1. L'utilisateur A se connecte au pMIX et obtient un emplacement.
2. A envoie l'index de son emplacement à B lors d'un tour d'appel.
3. L'utilisateur B se connecte au pMIX et obtient un emplacement.
4. B envoie l'index de son emplacement à A lors d'un tour d'appel.
```

Les étapes un et trois ne sont exécutées que si les utilisateurs ne sont pas déjà connectés au pMIX. Après ce protocole, les utilisateurs envoient des requêtes PIR pour l'emplacement de leur interlocuteur au lieu d'envoyer des requêtes aléatoires. Ainsi, à chaque tour *A* et *B* émettent en écrivant dans leur emplacement et obtiennent le contenu de l'emplacement de l'autre en l'extrayant de la réponse PIR renvoyée par le pMIX.

4.1.1.2 Anonymat de communication

Pendant la communication, si la sécurité du protocole PIR est assurée, aucun attaquant (même un administrateur du pMIX), ne peut savoir quels emplacements sont lus par quels utilisateurs. Aucun attaquant ne peut donc savoir quel est le récepteur d'un message écrit dans un emplacement. Ainsi, les messages entrant et sortant (les réponses PIR) ne peuvent pas être associés directement, comme c'est le cas dans un relais traditionnel.

4.1.1.3 Réduction du nombre de requêtes PIR

Une base de données peut utiliser une même requête PIR pour générer plusieurs réponses quand son contenu évolue. Ainsi, dans le cas du pMIX, un utilisateur s'intéressant à un emplacement donné n'a pas besoin d'envoyer une requête PIR à chaque tour. L'envoi d'une nouvelle requête PIR n'est d'ailleurs nécessaire que

lorsqu'un utilisateur veut changer l'emplacement à partir duquel il reçoit des données, c'est-à-dire quand il change d'interlocuteur. Les changements d'interlocuteur se faisant lors des tours d'appel, il est donc raisonnable que chaque utilisateur envoie une requête PIR à chaque tour d'appel, au lieu d'à chaque tour de communication. Ceci permet de réduire fortement le facteur d'expansion en émission pour les utilisateurs, sans nuire à l'anonymat puisque tous les utilisateurs continuent à agir de façon identique du point de vue du pMIX, ni à l'interactivité puisque les changements d'interlocuteur ne peuvent se faire qu'aux tours d'appel, indépendemment de la fréquence d'envoi des requêtes PIR. Si plusieurs tours d'appel se produisent avant qu'une communication entre deux utilisateurs se termine, ceux-ci enverront à chaque tour d'appel une nouvelle requête PIR pour l'index de leur interlocuteur.

4.1.1.4 Version finale

Dans la suite de cette section nous donnons une description du pMIX qui est illustrée par les figures 4.1, 4.2 et 4.3 où est présenté un exemple d'une communication entre deux utilisateurs. Les schémas à gauche représentent des messages envoyés pour la connexion au pMIX ainsi que pour les tours d'appel, et les schémas à droite représentent les écritures dans les emplacements et les réponses PIR. Nous avons séparé ces deux types de messages pour avoir des schémas plus clairs mais bien évidemment leurs émissions sont entrelacées.

Un utilisateur A qui veut se connecter au pMIX commence par envoyer une demande de connexion (schéma a de la figure 4.1). Quand il reçoit du pMIX l'index de l'emplacement qui lui est alloué, l'utilisateur A attend le prochain tour d'appel(schéma a). Ces tours sont réalisés par l'ensemble des utilisateurs connectés avec une période fixe τ_{APP}. Le premier message d'un tour d'appel est l'envoi de bourrage chiffré ou la participation au protocole d'envoi superposé mis en place pour l'envoi anonyme des index des emplacements (légende dans le coin supérieur gauche de la figure). Après un délai déterminé par le temps moyen que met un utilisateur pour recevoir un index et se connecter au pMIX, l'utilisateur envoie une requête PIR aléatoire.

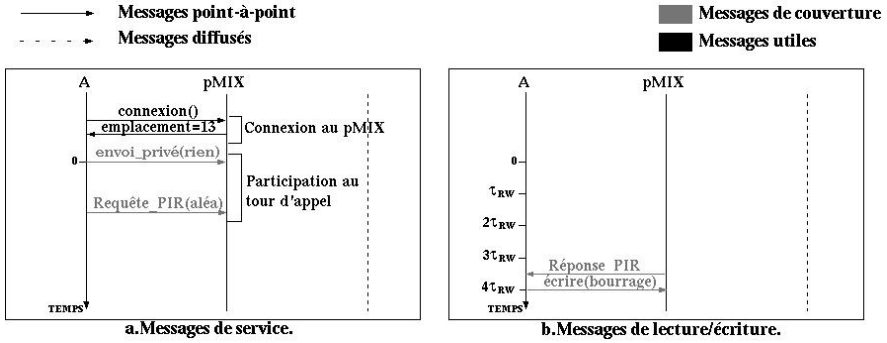


FIG. 4.1: Connexion au pMIX et attente.

À la fin du premier tour d'appel, le pMIX dispose d'une requête PIR de l'utilisateur *A*, et peut lui envoyer des réponses PIR basées sur le contenu des emplacements et sur la requête PIR qui lui est associée. À ce point, le pMIX considère que l'utilisateur *A* a terminé sa procédure de connexion, et commence à lui envoyer des réponses PIR à chaque tour de communication, ces tours étant réalisés tous les τ_{RW}. Après avoir reçu la première réponse PIR, l'utilisateur *A* écrit du bourrage chiffré dans son emplacement à chaque tour de communication (schéma b). Le pMIX fixe les valeurs de τ_{RW} et de τ_{EMPL}. en fonction de la latence du réseau, de ses capacités de traitement, de l'intéractivité souhaitée, et des application recherchées par les utilisateurs.

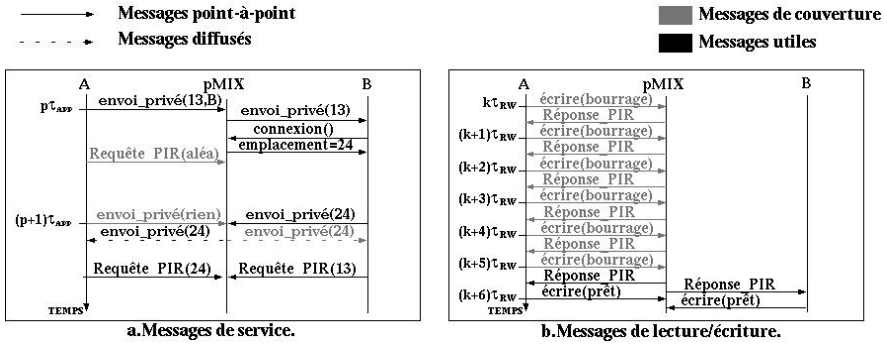


FIG. 4.2: Mise en place de la communication.

Quand un utilisateur *A* veut contacter un utilisateur *B* qui n'est pas connecté au pMIX il doit suivre le protocole indiqué ci-dessous et illustré par la figure 4.2.

Protocole 4.2 Protocole de mise en place d'une communication (version finale)

1. A envoie l'index de son emplacement à B lors d'un tour d'appel.
2. L'utilisateur B se connecte au pMIX et obtient un emplacement.
3. Au premier tour d'appel possible, B envoie l'index de son emplacement à A et une requête PIR pour l'index de l'emplacement de A.
4. Au prochain tour d'appel, A envoie au pMIX une requête PIR pour l'index de l'emplacement de B.

À la fin du protocole, la connexion de B est finalisée et le pMIX lui envoie des réponses PIR (à partir desquelles B pourra obtenir le contenu de l'emplacement de A) tous les τ_{RW}. B agit comme tout utilisateur se connectant au pMIX, et après avoir reçu la première réponse PIR, il commence à écrire dans son emplacement tous les τ_{RW} et à réaliser des tours d'appel tous les τ_{APP} (figure 4.3).

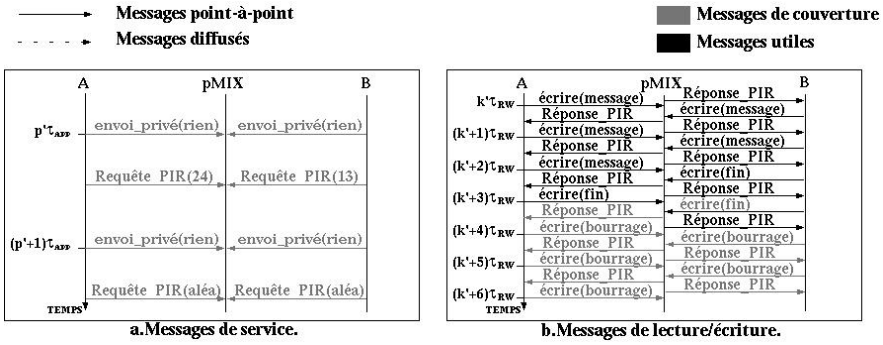


FIG. 4.3: Fin de communication.

Si l'utilisateur A veut contacter un utilisateur B qui est déjà connecté au pMIX le protocole reste inchangé, mis à part que la deuxième étape n'est pas réalisée.

4.1.1.5 Attaques actives des administrateurs

Les administrateurs du pMIX peuvent corrompre le contenu des emplacements, donner des réponses PIR erronées ou réaliser d'autres actions afin d'obtenir des informations par les différentes réactions que ceci provoque chez les utilisateurs. Pour ces raisons, les utilisateurs doivent échanger des messages authentifiés, et en

cas de dysfonctionnement ne pas se déconnecter ou agir d'une façon révélatrice. Si le dysfonctionnement persiste, l'utilisateur attendra le temps qu'il pense que ses transactions allaient durer avant de se déconnecter. Un pMIX avec un tel comportement ne tardera pas à être considéré comme malicieux ou peu fiable et sera rejeté par ses utilisateurs. Afin de détecter ce genre de comportements des envois superposés peuvent par exemple être faits de façon distribuée par les utilisateurs en fin de journée pour évaluer anonymement le pMIX.

4.1.2 Performances

Les performances du pMIX dépendent principalement du protocole PIR utilisé, et des paramètres qui sont choisis pour celui-ci. Tous les protocoles PIR présentés dans le deuxième chapitre ont un coût de traitement comparable. Étant donné que dans la plupart des applications on aura $\tau_{APP} \gg \tau_{RW}$, la taille des requêtes sera en général un paramètre moins important que le facteur d'expansion des réponses, ce pourquoi le protocole de Lipmaa nous semble le plus adapté aux pMIX.

Que l'envoi des index se fasse à travers un serveur de type EBBS ou DC-net, les résultats sont similaires, puisqu'à la fois le coût de traitement et celui des communications sont très majoritairement dus au protocole PIR. Nous présentons dans la section suivante l'analyse de performances d'un pMIX qui utilise le protocole de Lipmaa et un service de type DC-net pour les tours d'appel.

4.1.2.1 Bande passante utilisable et taille des groupes non-observables

La complexité des traitements est un problème majeur pour les protocoles PIR car la même opération doit être exécutée sur chacun des bits composant la base de données afin de ne pas donner d'information aux administrateurs sur les éléments intéressant un utilisateur. Le coût pour générer une réponse PIR est par conséquent linéaire avec la taille de la base. Dans le cas du pMIX ceci veut dire que pour générer une requête PIR, la base aura un coût de traitement $C_{REP} = C_{PIR} \times n \times l$, où l est la taille en bits d'un emplacement, n le nombre de clients connectés au pMIX, et C_{PIR} le coût de traitement du protocole PIR par bit dans une base. Pour tous les protocoles que nous avons présenté au chapitre 3, C_{PIR} est approximativement le coût d'une multiplication modulaire. Dans le cas du protocole de Lipmaa, le module a pour taille $(s + 1) \times k$ bits en gardant les notations de la section 3.5.

Le pMIX doit calculer un certain nombre de réponses PIR par seconde, qui lui aussi augmente linéairement avec le nombre d'utilisateurs. De chaque réponse PIR, l'utilisateur peut extraire l bits d'information (la taille d'un emplacement), et donc en notant C_{Comp} les capacités de traitement du pMIX, et B la bande passante utilisable par utilisateur on aura :

$$B = \frac{C_{Comp} \times l}{C_{REP} \times n} = \frac{C_{Comp}}{C_{PIR} \times n^2}$$

Étant donné que les utilisateurs écrivent et lisent à la même fréquence, la bande passante utilisable est la même dans les deux sens. Celle-ci est inversement proportionnelle à n^2, n étant le nombre d'utilisateurs connectés au pMIX. Le facteur d'échelle d'un tel système est donc limitée.

4.1.2.2 Coût des communications

Qu'un utilisateur soit vraiment en train de communiquer ou pas, les faux changements d'emplacement et l'envoi de bourrage chiffré assurent que la largeur de bande occupée soit constante. Nous distinguons deux sortes de trafic : le trafic d'information, qui peut être de l'information ou du bourrage, et le trafic de service qui est composé des requêtes PIR, et des envois et réceptions d'index.

Le temps qu'un utilisateur doit attendre entre le moment où il décide de commencer une communication et le moment où il peut le faire est d'en moyenne $\tau_{APP}/2$. Cette valeur peut être de par exemple cinq secondes pour des communications de voix sur IP, ou de deux secondes pour la navigation HTTP. Indépendemment de ce paramètre, il est important pour étudier les performances de savoir avec quelle fréquence les utilisateurs établissent des communications. Par exemple, un utilisateur moyen n'aura probablement pas plus de dix communications de voix sur IP dans la journée et même si τ_{APP} est de l'ordre de 10 secondes, le temps moyen entre deux établissements de connexion pour un utilisateur sera de l'ordre d'une heure. Nous notons ce temps moyen τ_{IND}. Pour n utilisateurs, il y aura en moyenne, $\tau_{IND}^{-1} \times n$ utilisateurs qui enverront un index par unité de temps, et il faudra un tour d'envoi superposé pour chacun d'entre eux. Ainsi, si les index sont chiffrés sur 1024 bits chaque utilisateur occupera en moyenne une largeur de bande passante $Band_{Index} = 1024 \times \tau_{IND}^{-1} \times n$ en émission et en réception.

Pour le protocole PIR de Lipmaa, afin de réduire au maximum le coût de traitement et le facteur d'expansion des réponses, nous prenons $s = L = 1$, et donc la taille d'une requête PIR est $Q_{PIR} = 2048 \times n$. La largeur de bande occupée par les requêtes PIR est donc $Band_{PIR} = 2048 \times \tau_{APP}^{-1} \times n$ en émission, et le trafic de service occupera une largeur de bande :

$$\begin{aligned} Band_{SE} &= 2048 \times \tau_{APP}^{-1} \times n + 1024 \times \tau_{IND}^{-1} \times n \\ Band_{SR} &= 1024 \times \tau_{IND}^{-1} \times n \end{aligned}$$

respectivement en émission et en réception. D'un autre côté, le facteur d'expansion des réponses PIR, pour le protocole et les paramètres choisis, est de deux, et la largeur de bande occupée pour le trafic d'information sera de :

$$\begin{aligned} Band_{IE} &= B \\ Band_{IR} &= 2 \times B \end{aligned}$$

respectivement en émission et en réception. En ajoutant les largeurs de bande occupée par l'information et par le trafic de service nous obtenons les largeurs de bande totales, à savoir :

$$\begin{aligned} Band_{E} &= B + 2048 \times \tau_{APP}^{-1} \times n + 1024 \times \tau_{IND}^{-1} \times n \\ Band_{R} &= 2 \times B + 1024 \times \tau_{IND}^{-1} \times n \end{aligned}$$

4.1.2.3 Exemple

Nous présentons ici des estimations de la performance attendue pour un pMIX sur un serveur bi-Opteron 248. Nous supposons que la seule limite est la puissance de traitement du pMIX. Les résultats montrent que c'est bien probablement la seule contrainte à prendre en compte dans la plupart des cas (sur un réseau local comme sur Internet).

La figure 4.4 présente la bande passante utilisable pour chaque utilisateur B et la largeur de bande occupée en émission et en réception pour un nombre d'utilisateurs allant de 7 à 25, avec $\tau_{IND} = 20$ minutes, et $\tau_{APP} = 10$ secondes, ainsi

que le facteur d'expansion des communications (largeur de bande occupée / bande passante utilisable) et la latence du système dans un réseau local. Pour cette valeur de τ_{IND}, l'impact de l'envoi des index sur les émissions est pratiquement inobservable. Même si nous supposons qu'en moyenne un utilisateur entame une nouvelle communication toutes les minutes, ceci n'augmente les émissions que de un demi-kilobit par seconde pour une trentaine d'utilisateurs connectés. Dans la plupart des applications l'impact sur le coût en communication de l'émission des index est négligeable.

4.1.2.4 Évaluation

Il est important de remarquer que la bande passante utilisable est proportionnelle à la puissance de traitement. De plus, la génération des réponses PIR est facile à distribuer et donc pour obtenir suffisamment de bande passante pour une application donnée, il suffit de multiplier le nombre de serveurs exécutant les calculs. Cependant les très faibles valeurs de la bande passante utilisable pour des groupes au-delà d'une vingtaine de membres laissent deviner que mettre en place un pMIX peut se révéler extrêmement cher pour beaucoup d'applications, si les groupes non-observables ne sont pas de petite taille.

Quand la bande passante utilisable est très petite, l'importance relative des requêtes PIR envoyées par les utilisateurs grandit fortement. Nous insistons sur le fait que ceci n'est pas tellement dû à la taille du groupe ni donc au pMIX en soi, mais au fait que les protocoles PIR actuellement disponibles nécessitent une puissance de calcul importante[1], et donnent lieu à des petites bandes passantes. L'importance relative des requêtes PIR augmente d'autant plus que τ_{APP} est petit, et donc il est important de choisir un équilibre entre le temps d'attente pour l'établissement des communications et le facteur d'expansion dans les émissions.

[1] Avant que le cryptosystème de Domingo-Ferrer [DF96] ne soit cryptanalysé nous disposions de protocoles PIR beaucoup plus performants donnant lieu a des débits de l'ordre de mille fois plus importants [AD05].

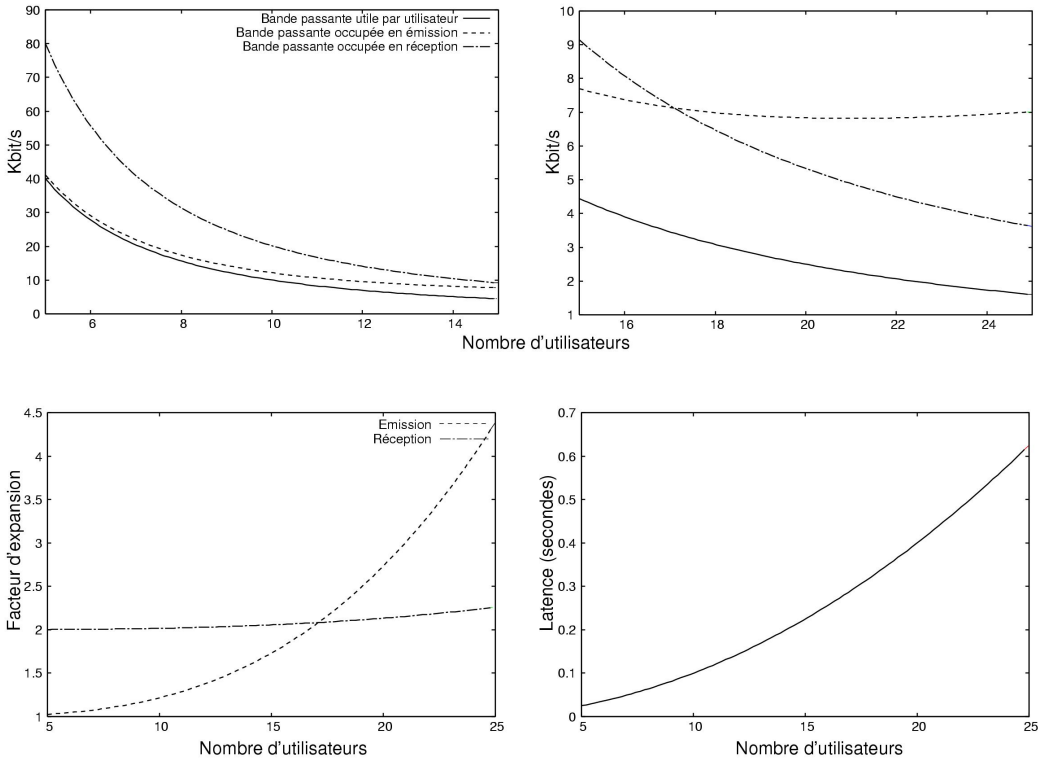


FIG. 4.4: Performances du pMIX.

La latence introduite par le calcul des réponses PIR varie de quelques millisecondes à 600 millisecondes. Cette latence, à nouveau, peut être divisée par un facteur quelconque si le calcul des réponses est distribué entre plusieurs serveurs. De même que pour la bande passante utilisable, les valeurs de la latence indiquent que, pour obtenir des petites latences avec un coût raisonnable, il faut que le groupes non-observables soient de très petite taille (de l'ordre de la dizaine d'utilisateurs).

4.2 Les variantes du pMIX

Les pMIX nécessitent une puissance de traitement importante, même quand ils ne fournissent qu'une petite bande passante à des petits groupes d'utilisateurs. Trois faits sont à l'origine de ce coût. Premièrement, le coût par bit dans une base de

données des protocoles PIR actuels est très important. Deuxièmement, le nombre de réponses PIR à calculer par seconde est proportionnel au nombre d'utilisateurs connectés. Et troisièmement, le coût pour générer une réponse PIR est lui aussi proportionnel au nombre d'utilisateurs connectés au pMIX.

L'obtention de protocoles PIR avec un faible facteur d'expansion dans les communications et un coût de traitement le plus bas possible est un domaine de recherche à part entière. Même si le but de ce travail n'est pas de trouver un protocole plus efficace que ceux déjà existants, il faut remarquer que les pMIX montrent à quel point des éventuelles avancées dans le domaine des protocoles PIR peuvent avoir un impact important dans le domaine des communications anonymes.

Dans la section 4.2.1 nous présentons une variante du pMIX qui permet de calculer un nombre de réponses PIR par seconde proportionnel uniquement au nombre de communications qui ont simultanément lieu, et non au nombre d'utilisateurs connectés au serveur [AD06]. Dans la section 4.2.2 nous présentons une variante du pMIX où le coût pour générer une réponse PIR n'est plus proportionnel au nombre d'utilisateurs connectés mais au nombre de communications en cours.

4.2.1 Le apMIX

4.2.1.1 Principe

Le pMIX décrit dans la section 4.1.1 doit calculer $1/\tau_{RW}$ réponses PIR par utilisateur et par seconde. Beaucoup d'utilisateurs peuvent ne pas être en train de communiquer à chaque instant, et ainsi la plupart des réponses PIR ne seront pas lues. Pour réduire le coût de traitement, le pMIX ne devrait calculer des réponses PIR que pour les utilisateurs qui sont réellement en train de communiquer. Cependant, si seuls les utilisateurs qui communiquent envoyaient des requêtes PIR, les administrateurs du pMIX pourraient savoir quels utilisateurs communiquent et lesquels ne communiquent pas.

Pour éviter qu'un administrateur du pMIX puisse identifier qui envoie des requêtes PIR et qui n'en envoie pas, les utilisateurs peuvent envoyer leurs requêtes PIR par un protocole d'envoi superposé, ce qui fournit une non-observabilité en émission. Si à un moment donné, il n'y a que m communications bidirectionnelles entre les utilisateurs, $2m$ utilisateurs envoient des requêtes PIR. Pour traiter les collisions, les utilisateurs peuvent utiliser une technique de réception superposée (cf. 2.2.2.2)

et en 2*m* rounds du protocole d'envoi superposé, le pMIX obtient les 2*m* requêtes PIR générées par les utilisateurs communiquant. Dès lors, les administrateurs du pMIX sont incapables d'associer les requêtes à des utilisateurs particuliers (voir figure 4.5).

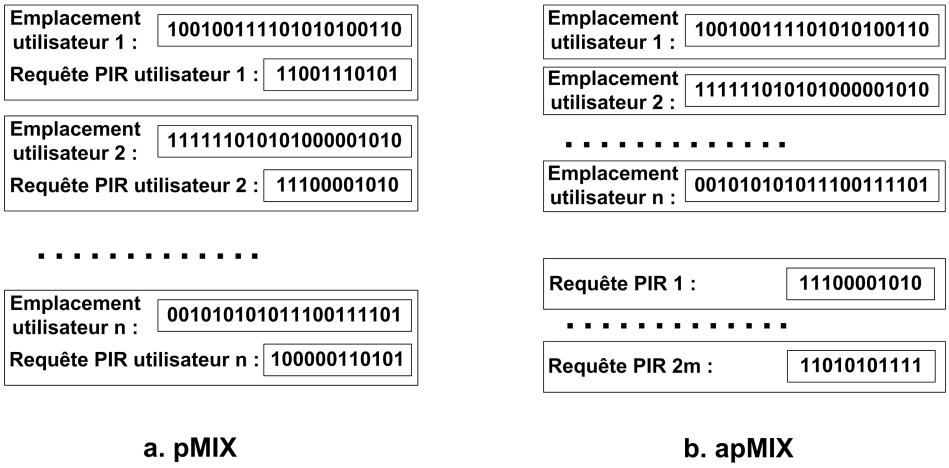


FIG. 4.5: Requêtes PIR anonymes.

Comme les requêtes PIR ne sont pas associables aux utilisateurs, il n'est pas possible pour le pMIX de savoir qui est intéressé par quelle réponse PIR générée. Étant donné que le pMIX ne peut pas savoir à qui envoyer quelle requête, il peut par exemple, envoyer toutes les réponses PIR à tous les utilisateurs. Avec une telle stratégie, le coût de traitement se voit nettement réduit, au prix d'une augmentation du facteur d'expansion en réception des utilisateurs. Nous appelons un tel serveur un apMIX. Les facteurs d'expansion pour le trafic d'information sont respectivement de 1 et de $2m \times F_{PIR}$, en notant F_{PIR} le facteur d'expansion du protocole PIR utilisé. Le coût de traitement quant à lui est proportionnel à $n \times 2m$ à comparer à celui du pMIX qui l'est à n^2.

Dans un apMIX quand un utilisateur met fin à une communication le nombre de requêtes PIR issues du protocole d'envoi superposé décroît de deux. De même quand un utilisateur établit une communication, le nombre de requêtes PIR croît de deux. Il est facile pour le pMIX de cacher vis-à-vis des utilisateurs le nombre de requêtes PIR qui sont issues de l'envoi superposé, en demandant un nombre de tours d'envoi superposé constant (cf. 2.2.2.3). Cependant, les administrateurs

du pMIX non seulement savent combien de communications ont lieu, mais aussi quand sont établies et quand se terminent les communications, comme dans le cas des serveurs DC-net.

Les utilisateurs communicant peuvent continuer à réaliser périodiquement des faux changements d'emplacement, afin d'éviter que les administrateurs puissent savoir quels débuts et quelles fins de communication correspondent à une même communication. Cependant il est possible que les administrateurs arrivent à associer statistiquement les débuts et fins de communication en se basant sur des durées de communication caractéristiques.

4.2.1.2 Performances

Cette variante, tout comme les serveurs DC-net, est intéressante quand le nombre de communications simultanées est petit devant le nombre d'utilisateurs, c'est à dire quand $m \ll n$. La figure 4.6 présente les performances pour un apMIX dans les mêmes conditions que celles que nous avions considéré pour le pMIX. Afin de réduire le nombre de paramètres, nous prenons pour abscisse la valeur $n \times 2m$. Ainsi, par exemple, l'abscisse $n \times 2m = 100$ représente les performances que peut atteindre un système où il y a une communication pour cinquante utilisateurs connectés au pMIX, ou bien deux communications pour vingt-cinq utilisateurs connectés.

N'ayant pas de faux changement d'emplacement, le seul trafic significatif en émission est celui d'information, et le facteur d'expansion est donc de un. En réception le seul trafic est aussi celui d'information, et le facteur d'expansion est celui du protocole PIR utilisé. Pour ces raisons, nous ne présentons dans cette figure que l'évaluation de la bande passante disponible par utilisateur et de la latence.

Les résultats de la figure 4.6 montrent que dans le cas de la voix sur IP notre serveur permettrait de gérer une communication anonyme au sein d'un groupe de jusqu'à environ cinquante utilisateurs. Pour supporter plus d'utilisateurs, de débit, ou de communications il suffit d'augmenter la puissance de calcul linéairement en fonction de ces paramètres. Ainsi, par exemple, une grappe de quatre serveurs apMIX bi-Opteron 248 permet par exemple d'avoir jusqu'à deux communications simultanées dans un réseau de cent postes.

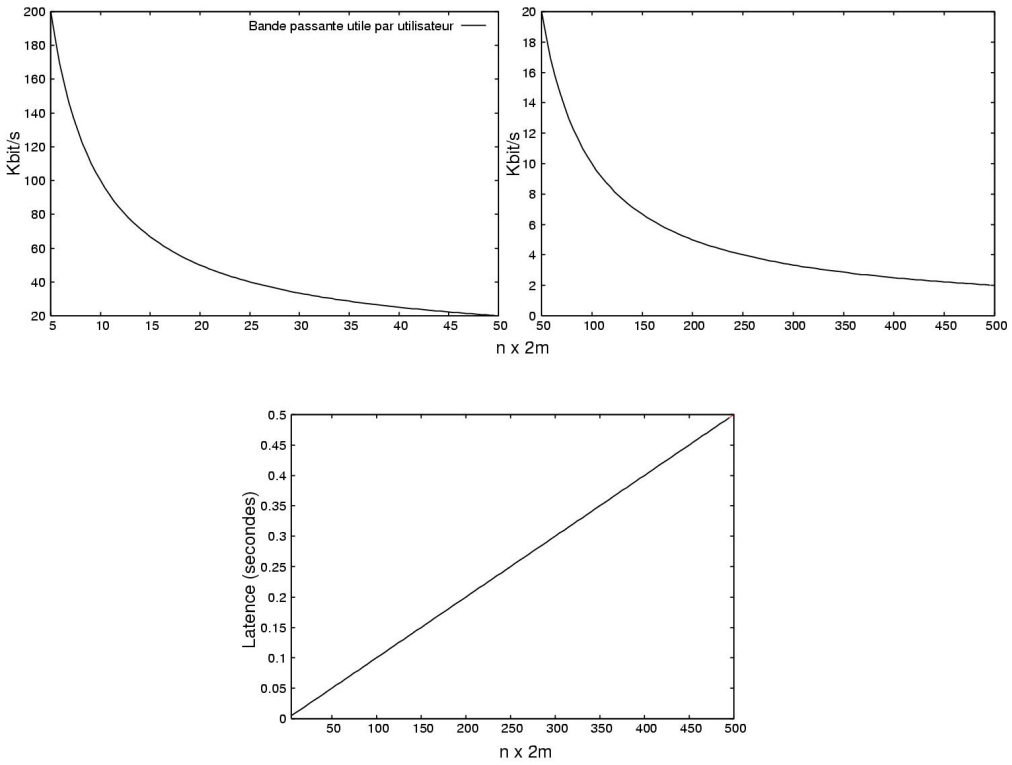


FIG. 4.6: Performances de l'apMIX.

Nous tenons a remarquer que ceci est indépendant du contexte des utilisateurs, ceux-ci peuvent être locaux ou avoir une connexion ADSL ou UMTS : les émissions n'ayant pas de facteur d'expansion cette alternative est bien adaptée aux lignes asymétriques. La bande passante en réception n'étant proportionnelle qu'au nombre de communications simultanées, en principe on peut avoir des groupes anonymes de très grande taille. Dans le cas de l'UMTS, il faut cependant remarquer que la voix sur IP demandant une bande passante considérable par rapport à celle fournie par cette technologie, les limites que nous nous sommes fixées en réception imposent que le nombre de communications simultanées ne dépasse pas deux.

Dans un apMIX, on utilise le bourrage chiffré pour couvrir les émissions des utilisateurs. L'envoi superposé n'est utilisé que pour publier les requêtes PIR et ceci est réalisé qu'une fois toutes les quelques secondes. Les fluctuations de la latence des connexions Internet sont très inférieures à cet ordre de grandeur, et donc la lo-

calisation des utilisateurs sur Internet n'est pas problématique, alors que serait le cas si on utilisait l'envoi superposé pour transmettre les messages. De même, une déconnexion impromptue d'un utilisateur sera probablement repérée avant que le tour d'envoi superposé n'ait lieu, et si ce n'est pas le cas, cela n'aura pas d'impact sur les communications déjà en cours.

4.2.2 Les serveurs pDC-net et apDC-net

4.2.2.1 Principe

Dans un pMIX, les requêtes PIR sont associées aux utilisateurs qui les envoient. Pour cette raison, afin d'obtenir une non-observabilité de réception vis-à-vis des administrateurs du pMIX, chaque utilisateur doit envoyer des fausses requêtes PIR quand il ne communique pas. Nous avons vu dans la section précédente que les utilisateurs peuvent envoyer les requêtes PIR par un protocole d'envoi superposé, et ainsi réduire le nombre de requêtes PIR à deux fois le nombre de communications réelles.

Dans cette section, au lieu de rendre les utilisateurs non-associables aux requêtes PIR, nous utilisons l'envoi superposé pour rendre les utilisateurs non-associables à leurs emplacements. Ceci correspond au quatrième choix possible quand on considère qu'il y a deux moyens pour émettre de façon non-observable et deux pour recevoir de façon non-observable. L'EBBS utilise du bourrage et de la diffusion, le serveur DC-net utilise l'envoi superposé et la diffusion, le pMIX utilise le bourrage et le PIR, et ici nous utilisons l'envoi superposé et le PIR.

Dans cette variante, les utilisateurs émettent vers le serveur comme il le feraient vers un serveur DC-net avec réservation de canaux (cf. 2.2.2.1). A chaque période τ_{RW} les utilisateurs envoient leurs messages pour chacun des canaux actifs, et le serveur récupère les résultats du tour d'envoi superposé pour chacun des canaux. Le serveur écrit ensuite chacun des résultats dans un emplacement, chaque canal ayant un emplacement unique réservé. Grâce à l'envoi superposé, les canaux peuvent être bidirectionnels (cf. 2.2.2.2) et, seulement m canaux sont nécessaires pour les m communications.

Un tel serveur est très proche d'un serveur DC-net, mis à part que les utilisateur envoient des requêtes PIR pour la lecture, comme dans le cas des pMIX. Pour cette raison, nous appelons un tel serveur un serveur *pDC-net*. Les facteurs d'expansion

en émission et en réception pour le trafic d'information sont respectivement de m et de F_{PIR}, et le coût de traitement est proportionnel à $n \times m$.

Le nombre de tours d'envoi superposé que les utilisateurs doivent réaliser par τ_{RW} dévoile m et les utilisateurs peuvent déduire de ses variations quand commence une communication et quand elle se termine. Ceci est vrai pour un serveur pDC-net, tout comme pour un serveur DC-net. À nouveau, le serveur peut toujours occulter ceci en demandant aux utilisateurs de réaliser des tours fictifs, comme expliqué dans la section 2.2.2.3. Cependant, si pour l'apMIX occulter m en demandant aux utilisateurs d'exécuter plus de tours pour l'envoi de requêtes PIR augmente légèrement le trafic en émission, dans le cas des serveurs DC-net et pDC-net, si on veut occulter auprès des utilisateurs les variations de m jusqu'à une certaine valeur m_{MAX}, le facteur d'expansion en émission se verra augmenté de $m_{MAX} - m$, ce qui peut être une augmentation considérable.

Les requêtes PIR peuvent aussi être envoyées à travers un protocole d'envoi superposé comme pour les apMIX. Un tel serveur sera appelé un serveur *apDC-net*. Remarquons qu'un tel système aura des facteurs d'expansion de m et de $m \times F_{PIR}$ pour les émissions et les réceptions respectivement, ce qui rend le serveur plus complexe, plus coûteux en puissance de calcul, et conduit à des performances en communication moins bonnes qu'un simple serveur DC-net. Nous ne l'incluons dans notre description que pour présenter l'ensemble des combinaisons possibles avec les différentes approches.

4.2.2.2 Performances

Comme pour le apMIX, un serveur pDC-net n'a d'intérêt que si $m \ll n$. Par contre à différence d'un apMIX, les requêtes PIR sont associées aux utilisateurs et il faut que les utilisateurs réalisent des faux changements d'emplacement. Ceci veut dire que la largeur de bande occupée en émission dépend de n, ce qui nous empêche d'exprimer les performances en fonction de $n \times m$ comme nous l'avons fait pour l'apMIX. Dans la figure 4.7 nous présentons les résultats pour $m = 1$. Pour $m \neq 1$ on peut déduire les performances avec une faible marge d'erreur à partir des résultats présentés sur la figure : il suffit de diviser la bande passante utile et la largeur de bande occupée en réception par m, et de multiplier le facteur d'expansion en émission et la latence par m.

Afin de réduire la taille des requêtes nous prenons comme paramètre $L = 1$ pour

le protocole de Lipmaa, quand $n < 85$, $L = 2$ quand $85 \leq n \leq 175$ et $L = 3$ sinon. Les points singuliers des courbes de la figure 4.7 correspondent aux changements de valeur de L. Ceci à pour conséquence une augmentation de la largeur de bande passante occupée en réception qui est compensée par la diminution de la largeur de bande passante occupée en émission afin de diminuer le coût total des communications. Ces choix sont propres aux paramètres choisis pour le système, en particulier à la valeur de τ_{APP}. Dans des contextes précis comme par exemple pour les utilisateurs ayant des largeurs de bande disponibles qui sont asymétriques, on ne visera pas à réduire le coût total des communications mais par exemple à respecter un rapport entre la largeur de bande occupée en émission et en réception.

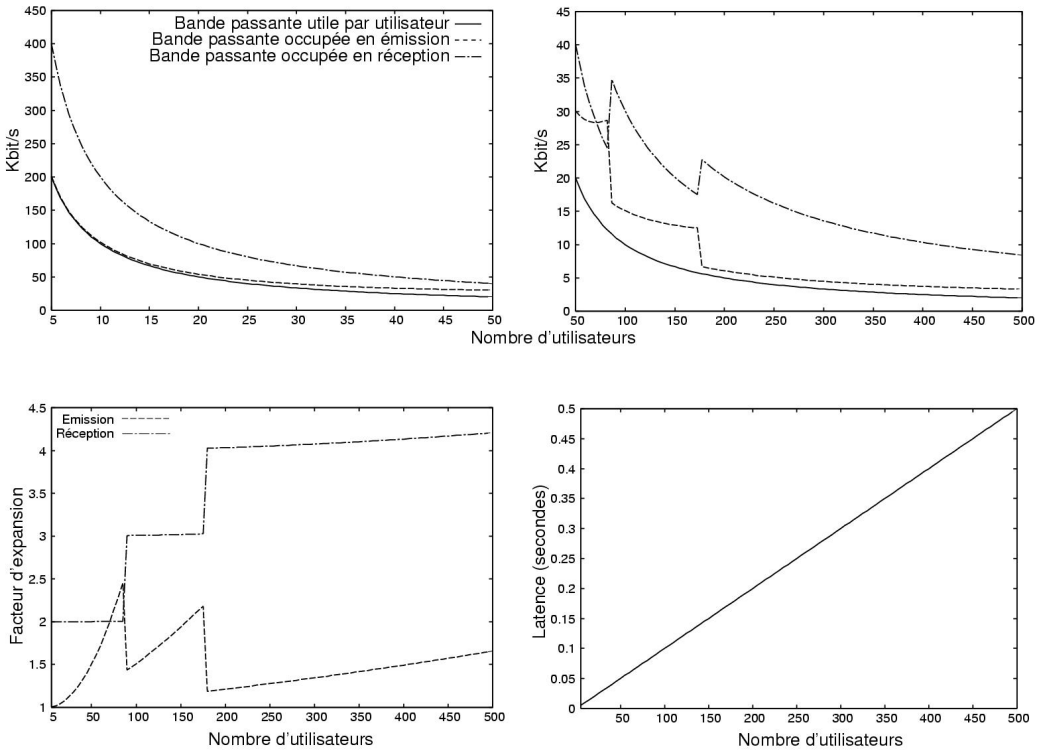


FIG. 4.7: Performances du serveur pDC-net.

Globalement les performances en bande passante disponible sont doublées par rapport à l'apMIX. Ceci vient du fait que dans un serveur pDC-net les canaux sont bidirectionnels, ce qui divise par deux le coût de traitement. Cependant tout comme les serveurs DC-net, cette variante demande une coordination importante

entre les utilisateurs (cf. 4.2.1.2), ce qui est à éviter quand les utilisateurs se connectent et déconnectent régulièrement, ou quand ils disposent d'une liaison peu fiable ou dont les performances sont variables.

4.3 Vue globale

De nos jours, deux approches sont habituellement citées comme moyens permettant de fournir un anonymat de communication : celle des serveurs DC-net et l'utilisation de relais multiples. Nous proposons ici une taxonomie dans laquelle l'utilisation de l'envoi superposé et de l'adressage implicite (c'est-à-dire l'approche des serveurs DC-net), peut être vue comme une instance dans un groupe de solutions, toutes dérivées des EBBS.

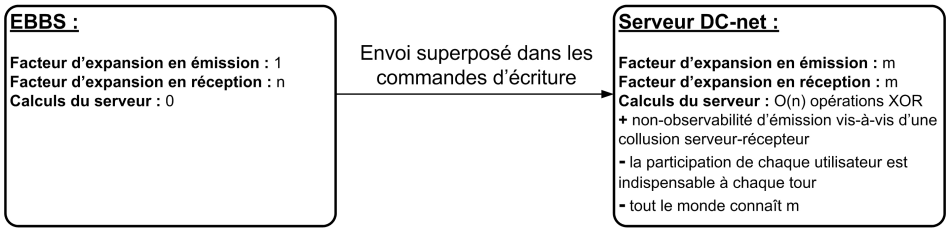


FIG. 4.8: Comparaison entre un EBBS et un serveur DC-net.

Un serveur DC-net peut être vu comme un EBBS dans lequel, afin d'écrire dans les emplacements de l'EBBS, les utilisateurs sont obligés de suivre un protocole d'envoi superposé (voir figure 4.8). En échange de quelques opérations au niveau du serveur (le XOR de tous les messages), on obtient certains avantages : le coût global des communications chute et les émetteurs sont non-observables, même vis-à-vis d'une collusion serveur-destinataire. Cette variante de l'EBBS a aussi certains inconvénients : m est connu publiquement, le facteur d'expansion en émission est m au lieu de 1, et la participation de chacun des utilisateurs est nécessaire à la terminaison d'un tour.

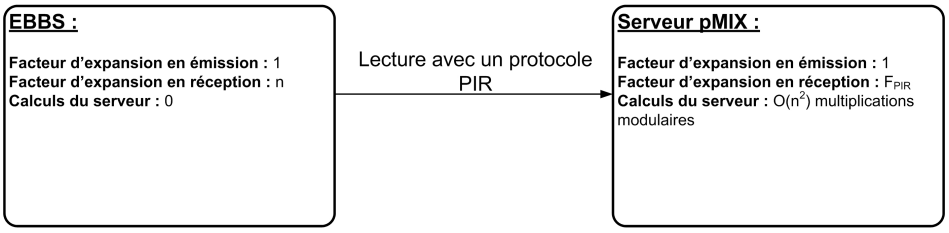


FIG. 4.9: Comparaison entre un EBBS et un pMIX.

D'un autre côté, un pMIX peut être vu comme un EBBS dans lequel au lieu d'utiliser la diffusion des messages, les utilisateurs réalisent des opérations de lecture à travers un protocole PIR (voir figure 4.9). Cette variante demande de grandes ressources de calcul mais le coût des communications est pratiquement optimal.

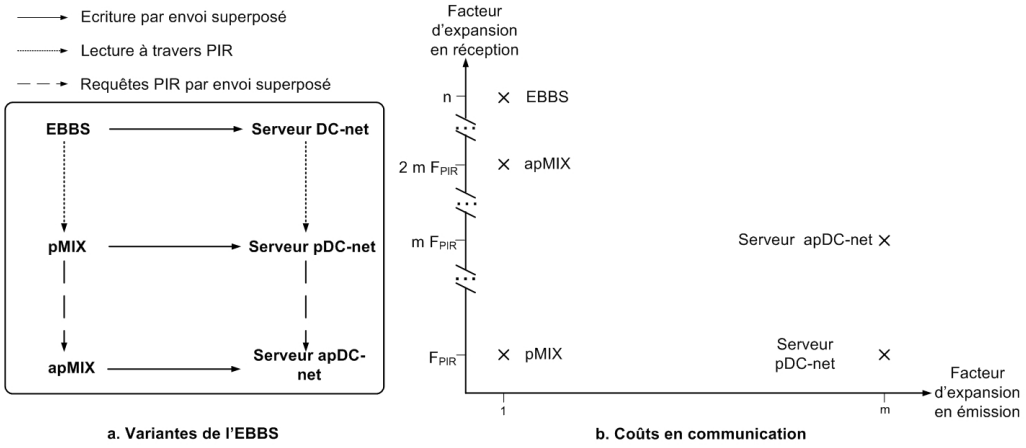


FIG. 4.10: Variantes de l'EBBS.

Le PIR et l'envoi superposé peuvent être utilisés tous les deux pour obtenir un serveur pDC-net. La publication des requêtes PIR peut elle même se faire en suivant un modèle de type EBBS, comme pour les serveurs déjà mentionnés, ou suivant un protocole d'envoi superposé, et dans ce cas on obtient les serveurs apMIX et apDC-net. La figure 4.10.a nous montre comment l'utilisation du PIR et de l'envoi superposé donne lieu à une constellation de variantes de l'EBBS initial.

Comparé à une approche classique qui serait de choisir entre les EBBS et les serveurs DC-net , la figure 4.10.b montre les différents choix en coût des émissions et des réceptions qui sont possibles pour concevoir des systèmes fournissant des communications anonymes, en considérant la combinaison du PIR aux autres techniques.

Conclusion générale

Dans ce mémoire, nous avons réalisé une étude des différentes approches possibles pour obtenir des communications anonymes à faible latence. Pour cela nous avons étudié différents systèmes avec des groupes d'utilisateurs contraints à être de petite taille, peu changeants, ou localisés. Notre démarche a été la suivante :

– Nous avons tout d'abord présenté les notions de base, et établi clairement quelles sont les primitives dont a priori on dispose pour anonymiser des communications, à savoir : l'utilisation de relais, l'émission par envoi superposé, l'émission de bourrage chiffré et la diffusion avec adressage implicite. Ensuite, nous avons mis en évidence que lorsqu'on considère un attaquant global ou suspicieux, il est nécessaire, dans le cadre des communications interactives, que le groupe des utilisateurs du service d'anonymisation soit non-observable en émission et en réception. Puis, nous avons exposé certains des problèmes fondamentaux qui limitent les performances des services existant aujourd'hui, et nous avons énoncé les contraintes auxquelles nous soumettons les groupes d'utilisateurs. Pour finir, nous remarquons que, quand les groupes d'utilisateurs sont soumis à ces contraintes, le risque d'attaquants globaux doit être pris en compte, ce qui implique que l'utilisation de relais ne convient pas à l'anonymisation des communications dans notre contexte.
– Après cette mise en place de la problématique, nous avons proposé deux types de serveurs basés sur les primitives disponibles : les EBBS, basés sur l'utilisation de bourrage chiffré en émission et de diffusion avec adressage implicite en réception, et les serveurs DC-net, basés sur l'utilisation de tours d'envoi superposé en émission et de diffusion avec adressage implicite en réception. Pour chacun de ces deux types de serveurs nous avons proposé des choix d'implémentation et des protocoles permettant d'optimiser leurs performances. L'évaluation des performances d'un EBBS étant simple, nous avons présenté un ensemble d'exemples d'utilisation en fonction des bandes passantes disponibles

en émission et en réception pour le serveur et pour les utilisateurs. Pour le serveur DC-net, nous avons réalisé une implémentation ainsi que des ensembles de tests de performance, dans une grappe et dans un réseau local de taille et de complexité importantes. De cette étude nous avons conclu que les serveurs DC-net représentent en général une meilleure solution que les EBBS dans le cadre d'un réseau local, notamment quand les initialisations et terminaisons de communications ne sont pas des informations sensibles qu'il faut occulter pour le serveur anonymisant les communications. La dissymétrie entre la bande passante en émission et en réception qui de nos jours caractérisent les connexions Internet, et la forte variance dans la qualité entre différentes connexions Internet et à différents moments pour une même connexion font que les serveurs DC-net ne sont pas un choix raisonnable quand les utilisateurs sont distribués sur l'Internet.

– Suite à cette étude, nous avons proposé de remplacer la diffusion avec adressage implicite par l'utilisation d'une primitive, la récupération d'informations privée (ou PIR). Cette primitive étant relativement méconnue, nous avons dédié un chapitre à part entière à sa présentation, dans le but d'éclairer le lecteur sur les subtilités de son utilisation, la force des preuves de sécurité assurant l'anonymat des utilisateurs, et les coûts de calcul inhérents. Premièrement, nous avons introduit les notions de base, puis nous avons pris distance par rapport aux définitions et aux critères d'évaluation standard afin de rendre la comparaison entre les différents protocoles PIR plus simple et faciliter le choix du meilleur protocole pour une application donnée. Ensuite, nous avons réalisé un historique des différents protocoles et indiqué les différentes avancées apportées par chaque nouveau protocole, en commençant par l'utilisation de prédicats à trappe ayant des propriétés homomorphiques et les protocoles récursifs, puis l'utilisation de systèmes de chiffrement homomorphiques ou de familles de prédicats variables avec des requêtes fixes. Finalement, nous avons comparé les différents protocoles et montré qu'il est facile de choisir quel protocole utiliser en fonction de la taille des éléments de la base de données, du nombre de ces éléments, du nombre de requêtes qu'un utilisateur réalise pour une base de données et d'autres facteurs comme une éventuelle dissymétrie dans la bande passante disponible pour les utilisateurs, ou des limitations dans leur capacité de stockage. Nous avons conclu cette taxonomie des protocoles PIR en remarquant qu'un serveur relayant de l'information peut être vu comme une base de données évoluant très vite, et que dans ce cas un protocole PIR peut être utilisé pour obtenir des groupes non-observables en réception. L'ensemble des définitions formelles et des preuves de sécurité a été placé dans l'annexe A.

– Dans le quatrième et dernier chapitre de ce mémoire, nous avons introduit des serveurs anonymisant les communications où la primitive de diffusion avec adressage implicite est remplacée par l'utilisation du PIR. Premièrement nous avons modifié un EBBS par étapes jusqu'à obtenir un serveur optimisé que nous avons appelé pMIX. Ce serveur anonymise les communications des utilisateurs sans pratiquement aucun facteur d'expansion dans les communications (que ce soit en émission où en réception) pour les utilisateurs. Cependant, le coût en calcul d'un tel serveur est extrêmement important et ceci pour deux raisons : le nombre de réponses PIR que le serveur doit calculer est proportionnel au nombre d'utilisateurs connectés et le coût de calcul de chacune de ces requêtes est extrêmement important (et d'ailleurs lui aussi proportionnel au nombre d'utilisateurs connectés). Ceci fait que le nombre maximum d'utilisateurs pouvant être connectés à un serveur haut de gamme de nos jours soit limité à une dizaine d'utilisateurs par exemple, pour des communications de voix sur IP. Les calculs sont facilement parallélisables, mais le coût de calcul croît quadratiquement par rapport au nombre d'utilisateurs (le nombre de réponses croît linéairement et la complexité de chacune aussi), ce qui fait que le coût d'une telle solution explose pour des groupes dépassant quelques dizaines d'utilisateurs. Pour ces raisons nous avons proposé plusieurs variantes du pMIX, résultant de la combinaison des primitives classiques avec les protocoles PIR. Ces serveurs utilisent beaucoup moins de puissance de calcul par utilisateur pour un débit donné, et diversifient largement les choix possibles pour anonymiser des communications quand les utilisateurs se trouvent distribués sur Internet. Pour illustrer cela, nous avons conclu ce chapitre par un aperçu global des options disponibles pour implémenter de tels services et montré que l'introduction du PIR et sa combinaison avec d'autres primitives classiques permettent d'avoir une famille de solutions bien plus polyvalente que la simple utilisation des primitives classiques ne le laissait entrevoir.

En conclusion, dans un réseau local, vu les coûts de calcul pour les protocoles PIR actuels, il vaut mieux utiliser :

· soit un EBBS, si la bande passante fournie par utilisateur multipliée par le nombre d'utilisateurs est très inférieure à la bande passante du réseau,
· soit un serveur DC-net, si le nombre de communications simultanées est limité.

Ainsi, dans un réseau local, nous avons vu qu'un EBBS permet d'anonymiser les communications de voix sur IP dans un groupe allant jusqu'à une centaine d'utilisateurs, pour un nombre de communications simultanées illimité. Un serveur DC-net permet d'anonymiser les communications de plusieurs centaines d'utili-

sateurs, mais le nombre de communications simultanées est restreint à une ou deux dizaines.

Quand les utilisateurs sont distribués sur l'Internet, les variantes du pMIX représentent le meilleur choix. En particulier, un apMIX permet par exemple d'implémenter un service d'anonymisation des communications de voix sur IP pour un coût raisonnable. Avec les choix que nous avons affinés tout au long de ce mémoire nous évaluons un coût de cinq cents euros en matériel pour le serveur avec :

– une cinquantaine d'utilisateurs,
– un débit de 10 Kbits/s,
– une seule communication simultanée.

Pour multiplier le nombre d'utilisateurs, le débit, ou le nombre de communications simultanées, il suffit de multiplier le nombre de serveurs linéairement en ces facteurs. Par exemple, pour environ huit mille euros, il est possible de mettre en place un serveur gérant deux cents utilisateurs ayant jusqu'à quatre communications anonymes simultanées (le *Call Manager Express*, un PABX sur IP de Cisco gérant jusqu'à 140 utilisateurs coûte environ cinq mille euros). D'après une étude menée par le groupe Solucom auprès de 402 sociétés en France, cinquante pour cent d'entre elles envisagent d'externaliser leurs communications téléphoniques par une solution de voix sur IP [Jea06]. De tels services sont souvent facturés entre une et trois dizaines d'euros par mois et par poste, chacun de ces postes ayant un coût de l'ordre de la centaine d'euros. Le coût en matériel pour le serveur étant d'une dizaine d'euros par poste et par communication simultanée pour un apMIX, le surcoût lié à l'anonymisation reste très raisonnable.

Les perspectives pour notre travail sont variées. D'un côté il serait intéressant de développer des implémentations abouties et de réaliser des tests à grande échelle pour chacune des implémentations. D'un autre, l'optimisation des protocoles PIR aurait un impact important sur le pMIX et ses variantes.

Obtenir des protocoles PIR où les utilisateurs écrivant dans une base de données chiffrent leurs messages en suivant un schéma propre au protocole PIR, et où les réponses PIR correspondent au message en clair final pour l'utilisateur destinataire, permettrait d'élargir grandement le domaine d'application de ces protocoles. En particulier, dans le contexte de l'utilisation des protocoles PIR avec les pMIX, ceci permettrait de contacter des utilisateurs extérieurs pour qui la présence du pMIX serait transparente, ouvrant ainsi le domaine d'utilisation des pMIX à la navigation Web et à d'autres applications.

Les protocoles PIR actuels sont souvent basés sur des systèmes de chiffrement homomorphique. Ces systèmes de chiffrement reposent sur certaines hypothèses, et sont conçus pour être utilisés comme des systèmes à clé publique. Cependant, dans le cas du PIR, le chiffrement comme le déchiffrement sont réalisés par l'utilisateur qui génère la requête PIR. Il n'y a donc aucun besoin d'avoir un système à clé publique. Nous pensons que des nouveaux systèmes de chiffrement homomorphique, plus efficaces calculatoirement que les actuels, sont réalisables en introduisant la contrainte de maintenir les clés secrètes. De plus, certains cryptosystèmes considérés comme faibles, comme Polly-Cracker [SG02], peuvent trouver ici une utilisation, étant donné que l'ensemble de leurs cryptanalyses sont basées sur la connaissance de la clé publique. Le coût de calcul pour un protocole PIR basé sur un tel système peu être divisé par un facteur mille, facteur à distribuer entre le nombre d'utilisateurs maximum, le débit, et le nombre de communications simultanées. Avec une telle réduction du coût de calcul, le nombre d'applications pour les pMIX et leurs variantes sur l'Internet exploserait, et même dans les réseaux locaux ces systèmes représenteraient le meilleur choix pour l'anonymisation des communications.

Annexe A

Analyse formelle des protocoles PIR basés sur des systèmes de chiffrement homomorphiques

Beaucoup de publications sur les PIR opérant sur des bases de données non-répliquées ont vu le jour ces dernières années, en particulier sur les constructions basées sur des système de chiffrement homomorphique. Cependant, il est difficile de trouver des définitions formelles ainsi que des démonstrations rigoureuses dans le domaine. Dans cette annexe nous allons poser plusieurs définitions formelles de ces protocoles, et montrer leur équivalence. Ensuite, nous démontrons formellement que les protocoles basés sur la construction proposée par Stern vérifient les définitions formelles que nous avons données, et que ces protocoles sont généralisables, même si de nos jours les primitives dont on dispose ne nous permettent de le faire que de façon limitée.

A.1 Définitions

La plupart des notations et définitions proposées dans cette annexe sont celles que Goldwasser et Micali ont utilisées dans leur article fondateur [GM84].

Soit *MT* une machine de Turing probabiliste. On notera *MT[i]* l'espace des probabilités qui associe à la chaîne σ la probabilité de que la sortie de *MT* soit σ

pour une entrée i. Pour simplifier les notations, si MT dispose de plusieurs sorties, on gardera notation $MT[i]$ pour désigner un espace de probabilités associé à une sortie unique si la sortie nous intéressant est évidente. Si EP est un espace de probabilités, $x \leftarrow EP$ représente l'algorithme qui attribue à x la valeur $e \in EP$ avec probabilité $Pr_{EP}(e)$. Si E est un ensemble, $x \stackrel{P}{\leftarrow} E$ représente $x \leftarrow E_P$ où E_P est l'espace de probabilités défini par l'ensemble E et la distribution de probabilités P. Une distribution de probabilités notée U représentera toujours une distribution uniforme, par exemple $x \stackrel{U}{\leftarrow} \{0,1\}$ représente un lancer de pièce non-biaisé.

Enfin, si $f(.)$ et $g(.)$ sont deux algorithmes probabilistes, alors $f(g(.))$ sera l'algorithme obtenu par l'exécution de f sur la sortie de g.

A.1.1 Systèmes de chiffrement

Définition 1 *Un Générateur de Messages (GM) est une machine de Turing probabiliste à temps polynomial qui pour une entrée k, donne en sortie une chaîne de k bits qu'on appellera message.*

Définition 2 *Un Système de chiffrement est une machine de Turing probabiliste à temps polynomial* Π *qui pour une entrée k donne en sortie la description de deux algorithmes, E (l'algorithme de chiffrement) et D (l'algorithme de déchiffrement) tels que*

1. *pour une constante donnée c, E et D s'arrêtent en moins de* k^c *pas pour des entrées de taille k,*
2. *pour tout* $m \in \{0,1\}^k$, $D(E(m)) = m$.

La définition ci-dessus ne spécifie pas si le système de chiffrement est à clé publique ou à clé secrète. La raison principale pour ceci, est la séparation qui est normalement faite entre les définitions des systèmes de chiffrement, et leurs propriétés de sécurité. Un système de chiffrement à clé publique (qu'on notera PKC pour *Public Key Cryptosystem*), est un système de chiffrement dont les propriétés de sécurité sont prouvées quand E est connu. Dans le cas d'un système de chiffrement à clé secrète, les adversaires sont supposés juste connaître la description de la machine de Turing Π. Si les algorithmes de chiffrement issus de Π sont probabilistes on dira que le système de chiffrement est *randomisé*.

Définition 3 (PKC Sémantiquement sûr) *Soit* Π *un système de chiffrement randomisé, GM un générateur de messages, V un ensemble, et* $\mathcal{F} = \{f_E : GM[k] \rightarrow V | E \in \Pi[k], k \in \mathbb{N}\}$. *La probabilité de la valeur la plus probable de* $f_E(m)$ *est*

$$p_{f_E} = max\{\Sigma_{m \in f_E^{-1}(v)} Pr_{GM[k]}(m) | v \in V\},$$

c'est la probabilité maximum avec laquelle on peut deviner $f_E(m)$ *en connaissant juste la distribution de probabilités ayant donné lieu à m.*

Π *est un PKC sémantiquement sûr par rapport à GM, si quelle que soit la famille de fonctions* $\mathcal{F}$*, pour toute famille de circuits probabilistes de taille polynomiale* $A = \{A_k(\cdot, \cdot)\}$*, et pour tout* $c > 0$*, il y a un K tel que pour* $k > K$

$$Pr(A_k(E, \alpha) = f_E(m) | m \leftarrow GM[k]; E \leftarrow \Pi[k]; \alpha \leftarrow E(m)) < p_{f_E} + k^{-c}$$

On dira que Π *est un PKC sémantiquement sûr si il est sémantiquement sûr pour tout générateur de messages.*

La sécurité sémantique est dans un certain sens la même chose que la définition de la sécurité parfaite de Shannon si on ne considère que les adversaires ayant une puissance de calcul limitée polynomialement. Ceci est équivalent à d'autres définitions de sécurité comme par exemple la sécurité polynomiale (plus connue comme IND-CPA ou indistinguabilité) ou la sécurité-Y [MRS88], et est considéré comme l'un des degrés les plus hauts de sécurité qu'un système de chiffrement peut atteindre.

A.1.2 La récupération d'informations calculatoirement privée

Un protocole PIR est dit *calculatoirement privé* si deux propriétés sont respectées. Premièrement, un utilisateur doit pouvoir récupérer n'importe quel élément de la base de données. Deuxièmement, la base ne devrait pouvoir en un temps polynomial extraire d'une requête, aucune information sur quel élément intéresse l'utilisateur. La définition fondatrice [CG97] d'un PIR calculatoirement privé exprime ceci par l'indistinguabilité des requêtes des utilisateurs. En formalisant les notions d'indistinguabilité et de probabilité négligeable nous obtenons la définition suivante.

Définition 4 (Protocole PIR calculatoirement privé (1)) *Soit une base de données avec n éléments* $\{e_1, ..., e_n\}$*. Un protocole PIR calculatoirement privé est une machine de Turing probabiliste à temps polynomial* Γ_n*, qui pour une entrée k donne en sortie la description d'un algorithme randomisé à temps polynomial,* $Q(\cdot)$ *(le générateur de requêtes), et deux algorithmes déterministes à temps polynomial,* $R(\cdot,\cdot)$ *(le générateur de réponses), et* $X(\cdot,\cdot)$ *(l'algorithme d'extraction), tels que*

1. $X(R(Q(i), \{e_1, ..., e_n\}), \mathcal{S})$*,* $\mathcal{S}$ *étant toute information secrète nécessaire pour l'extraction, donne le i-ème élément de la base de données et,*
2. *pour tout* $c > 0$*, tout* $i_1, i_2 \in \{1, ..., n\}$*, et toute famille de circuits probabilistes de taille polynomiale* $C = \{C_k(\cdot,\cdot)\}$*, il y a un K tel que pour* $k > K$

$$Pr(C_k(\alpha, i_1, i_2, \mathcal{P}(k)) = i | i \overset{U}{\leftarrow} \{i_1, i_2\}; Q \leftarrow \Gamma_n[k]; \alpha \leftarrow Q(i)) < 1/2 + k^{-c}$$

où $\mathcal{P}(k)$ *est l'ensembles des informations publiques relatives au protocole PIR.*

Nous présentons ici une autre définition, que nous croyons plus claire, et qui est mieux adaptée aux preuves que nous donneront par la suite. Au lieu de spécifier que les requêtes doivent être indistinguables, nous exprimerons le fait qu'une base de données en possession d'une requête et d'une puissance de calcul polynomialement limitée, ne devrait pas augmenter significativement la probabilité de deviner quel élément l'utilisateur essaye de récupérer. Le lemme et le théorème suivant cette définition prouvent qu'elle est équivalente à la première définition que nous avons donnée.

La deuxième propriété dans cette définition est très proche de la notion de sécurité sémantique, de même que pour la première on s'approchait de la notion d'indistinguabilité pour un système de chiffrement. Il n'est donc pas étonnant que nos deux définitions soient équivalentes, tout comme le sont ces deux notions de sécurité dans les systèmes de chiffrement.

Définition 5 (Protocole PIR calculatoirement privé (2)) *Soient BDD une base de données avec n éléments* $\{e_1, ..., e_n\}$*, U un utilisateur de BDD, et P une distribution de probabilités sur* $\{1, ..., n\}$ *représentant par exemple les préférences de U*

pour le choix des éléments de BDD à récupérer. Pour tout $j \in \{1, ..., n\}$ définissons $p_j = Pr_P(j)$ et $p_M = max_{j\in\{1,...,n\}}\{p_j\}$. p_M est donc la probabilité maximale avec laquelle on peut a priori deviner quel sera l'index sur lequel portera une requête de l'utilisateur.

Un protocole PIR calculatoirement privé est une machine de Turing probabiliste à temps polynomial Γ_n, qui pour une entrée k donne en sortie la description d'un algorithme randomisé à temps polynomial, $Q(\cdot)$ (le générateur de requêtes), et deux algorithmes déterministes à temps polynomial, $R(\cdot, \cdot)$ (le générateur de réponses), et $X(\cdot, \cdot)$ (l'algorithme d'extraction), tels que

1. *$X(R(Q(i), \{e_1, ..., e_n\}), \mathcal{S})$, $\mathcal{S}$ étant toute information secrète nécessaire pour l'extraction, donne le i-ème élément de la base de données et,*
2. *pour tout $c > 0$, toute distribution de probabilités P, et toute famille de circuits probabilistes de taille polynomiale $C = \{C_k(\cdot, \cdot)\}$, il y a un K tel que pour $k > K$*

$$Pr(C_k(\alpha, \mathcal{P}(k)) = i | i \overset{P}{\leftarrow} \{1, ..., n\}; Q \leftarrow \Gamma_n[k]; \alpha \leftarrow Q(i)) < p_M + k^{-c}$$

où $\mathcal{P}(k)$ est l'ensembles des informations publiques relatives au protocole PIR.

Lemme 1 *Soit Γ_n un protocole respectant la définition 4, et $C = \{C_k(\cdot, \cdot)\}$ une famille de circuits probabilistes de taille polynomiale ayant leurs sorties dans $\{1, ..., n\}$. En gardant les notations utilisées dans la définition 4, on note $p_{i,j,k} = Pr(C_k(\alpha, \mathcal{P}(k)) = j | Q \leftarrow \Gamma_n[k]; \alpha \leftarrow Q(i))$. Pour tout $c > 0$ il y a un $K \in \mathbb{N}$ tel que pour tout $j \in \{1, ..., n\}$, tout $i_1, i_2 \in \{1, ..., n\}$, et tout $k > K$,*

$$|p(i_1, j, k) - p(i_2, j, k)| < k^{-c}$$

Preuve :

Supposons qu'il y ait un $c > 0$, $i_1, i_2, j \in \{1, ..., n\}$, une famille de circuits probabilistes de taille polynomiale C, et un sous-ensemble non-fini $N' \subseteq \mathbb{N}$ tel que pour tout $k \in N'$,

$$|p(i_1, j, k) - p(i_2, j, k)| > k^{-c}$$

On pose $p(i_{max}, j, k) = max\{p(i_1, j, k), p(i_2, j, k)\}$ et $p_j = 1/2(p(i_1, j, k) + p(i_2, j, k))$. Soit $D = \{D_k(\cdot, \cdot)\}$ une famille de circuits probabilistes de taille polynomiale qui

pour des entrées α et $\mathcal{P}(k)$, D_k appelle $C_k(\alpha, \mathcal{P}(k))$. Si C_k donne en sortie j, alors $D_k(\alpha, \mathcal{P}(k)) = i_{max}$, sinon D_k donne en sortie $x \overset{U}{\leftarrow} \{i_1, i_2\}$. On a donc,

$$P_{win} = Pr(D_k(\alpha, \mathcal{P}(k)) = i | i \overset{U}{\leftarrow} \{i_1, i_2\}; Q \leftarrow \Gamma_n[k]; \alpha \leftarrow Q(i)) = 1/2(1 - p_j) + p_j \times p_{max}/p_j$$

puisque p_j représente la probabilité de que C_k donne en sortie j, p_{max}/p_j la probabilité de succès si C_k donne en sortie j, et $1/2$ la probabilité de succès quand C_k ne donne pas en sortie j. Ainsi,

$$P_{win} = 1/2(1 + 1/2(p_{max} - p(i_1, j, k) + p_{max} - p(i_2, j, k))) > 1/2 + k^{-c}/4$$

On conclut donc que les requêtes pour l'index i_1 et les requêtes pour l'index i_2 sont indistinguables.

□

Theorème 1 *Les définitions 4 et 5 sont équivalentes.*

Preuve :

Non (1) implique Non (2).

Supposons que pour un protocole PIR Γ_n il y ait $c > 0$, une famille de circuits probabilistes de taille polynomiale $C = \{C_k(\cdot, \cdot)\}$, $i_1, i_2 \in \{1, ..., n\}$, et un sous-ensemble non-fini $N' \subseteq \mathbb{N}$ tel que pour tout $k \in N'$,

$$Pr(C_k(\alpha, i_1, i_2, \mathcal{P}(k)) = i | i \overset{U}{\leftarrow} \{i_1, i_2\}; Q \leftarrow \Gamma_n[k]; \alpha \leftarrow Q(i)) > 1/2 + k^{-c}$$

Si on considère la distribution de probabilités P telle qu'un utilisateur enverra une requête pour l'élément d'index i_1 avec une probabilité $1/2$, une requête pour l'élément d'index i_2 avec probabilité $1/2$, et une requête pour un élément ayant un autre index avec une probabilité nulle, la probabilité de que la base de données devine a priori sur quel index porte une requête sera $p_M = 1/2$ et donc pour c, N', et P, la famille des circuits probabilistes de taille polynomiale C est telle que pour tout $k \in N'$,

$$Pr(C_k(\alpha, \mathcal{P}(k)) = i | i \overset{P}{\leftarrow} \{1, ..., n\}; Q \leftarrow \Gamma_n[k]; \alpha \leftarrow Q(i)) > p_M + k^{-c}$$

Comme dans la définition 5, la deuxième propriété doit être respectée pour toute distribution P, on déduit donc que si la définition 4 n'est pas valable pour un protocole PIR donné, alors la définition 5 ne l'est pas non plus.

(1) implique (2).

Supposons que la définition 4 soit valable pour un protocole PIR Γ_n. Soit $C = \{C_k(\cdot,\cdot)\}$ une famille de circuits probabilistes de taille polynomiale qui ont leurs sorties dans $\{1, ..., n\}$.

Pour tout $c > 0$, toute distribution de probabilités P, et toute famille de circuits probabilistes à temps polynomial $C = \{C_k(\cdot,\cdot)\}$, on note

$$P_{win} = Pr(C_k(\alpha, \mathcal{P}(k)) = i | i \overset{P}{\leftarrow} \{1, ..., n\}; Q \leftarrow \Gamma_n[k]; \alpha \leftarrow Q(i))$$

Pour tout k, soit p_j pour tout $j \in \{1, ..., n\}$ la probabilité donnée à j par la distribution de probabilités P, et $p'_{j,k}$ la probabilité de succès pour le circuit C_k quand en entrée il a une requête pour un élément d'index j. On a donc pour tout k, $P_{win} = \Sigma_{j\in\{1,...,n\}} p'_{j,k} p_j$.

Soit $c > 0$. D'après le lemme 1, il y a un $K \in \mathbb{N}$ tel que pour tout $j \in \{1, ..., n\}$, tout $i_1, i_2 \in \{1, ..., n\}$, et tout $k > K$,

$$|p(i_1, j, k) - p(i_2, j, k)| < k^{-c}$$

ce qui implique que pour tout $i \in \{1, ..., n\}$, $|p(i_1, j, k) - p(i, j, k)| < k^{-c}$. Étant donné que $p'_{j,k} = p(j, j, k)$, on peut déduire que pour $k > K$,

$$|P_{win} - \Sigma_{j\in\{1,...,n\}} p(i_1, j, k) p_j| < k^{-c}$$

Étant donné que $\Sigma_{j\in\{1,...,n\}} p(i_1, j, k) = 1$, la somme $\Sigma_{j\in\{1,...,n\}} p(i_1, j, k) p_j$ est entre $min_{j\in\{1,...,n\}}\{p_j\}$ et $p_M = max_{j\in\{1,...,n\}}\{p_j\}$ et donc pour tout $k > K$

$$P_{win} = Pr(C_k(\alpha, \mathcal{P}(k)) = i | i \overset{P}{\leftarrow} \{1, ..., n\}; Q \leftarrow \Gamma_n[k]; \alpha \leftarrow Q(i)) < p_M + k^{-c}$$

□

A.1.3 Systèmes de chiffrement homomorphiques

Un système de chiffrement homomorphique, selon la définition donnée dans [BY87], est un système de chiffrement entre deux structures algébriques qui transforme certaines des opérations internes à la structure algébrique des textes clairs M, vers des opérations internes à la structure algébrique des textes chiffrés C. Nous sommes intéressés par deux cas de chiffrements homomorphiques. Premièrement ceux qui transforment les opérations entre deux groupes, puis ceux qui transforment les opérations entre deux anneaux.

Définition 6 *Pour un système de chiffrement* Π *et un générateur de messages* GM*, on note* $M = \{M_E = \{GM[k]\} | (E, D) \in \Pi[k], k \in \mathbb{N}\}$*, et* $C = \{C_E = E(M_E) | M_E \in M\}$*. On dit que* Π *est un système de chiffrement homomorphique de groupes, si pour tout générateur de messages* GM*, pour tout* $k \in \mathbb{N}$*, et pour tout* $(E, D) \in \Pi[k]$*,*

1. *il y a un ensemble d'opérations* $\{+_{M_E}, +_{C_E}\}$ *tel que* $(M_E, +_{M_E})$*, et* $(C_E, +_{C_E})$ *sont deux groupes et,*
2. *le résultat de l'addition, en utilisant* $+_{C_E}$ *de tout ensemble de textes chiffrés est déchiffré par D vers l'addition en utilisant* $+_{M_E}$ *des textes clairs associés.*

Définition 7 *Pour un système de chiffrement* Π *et un générateur de messages* GM*, on note* $M = \{M_E = \{GM[k]\} | (E, D) \in \Pi[k], k \in \mathbb{N}\}$*, et* $C = \{C_E = E(M_E) | M_E \in M\}$*. On dit que* Π *est un système de chiffrement homomorphique d'anneaux, si pour tout générateur de messages* GM*, pour tout* $k \in \mathbb{N}$*, et pour tout* $(E, D) \in \Pi[k]$*,*

1. *il y a un ensemble d'opérations* $\{+_{M_E}, \times_{M_E}, +_{C_E}, \times_{C_E}\}$ *tel que* $(M_E, +_{M_E}, \times_{M_E})$*, et* $(C_E, +_{C_E}, \times_{C_E})$ *sont deux anneaux et,*
2. *le résultat de tout ensemble d'opérations utilisant* $+_{C_E}$ *et* $\times_{C_E}$ *sur un ensemble quelconque de textes chiffrés est déchiffré par D vers le résultat donné par le même ensemble d'opérations utilisant* $+_{M_E}$ *et* $\times_{M_E}$ *sur les textes clairs associés.*

Remarque 1 *Dans un système de chiffrement homomorphique de groupes on peut dire que pour tout* $m \in M_E$ *et pour tout* $r \in \mathbb{N}$ *inférieur à l'ordre de m dans* M_E*, on a* $D(r \cdot_{C_E} \alpha) = r \cdot_{M_E} m$*, avec* $\alpha \in E(m)$ *et* $r \cdot_{M_E} m$ *représentant l'addition de r termes* $m +_{M_E} m +_{M_E} ... +_{M_E} m$ *(la multiplication scalaire).*

Remarque 2 *Quand la structure algébrique des opérations (multiplication scalaire, addition et multiplication internes) n'est pas équivoque, ces opérations seront notées sans spécifier la structure algébrique afin de simplifier les notations.*

On dira qu'un système de chiffrement homomorphique de groupes est pratique si les chiffrés n'augmentent pas de taille quand on les ajoute entre eux. Il existe beaucoup de système de chiffrement homomorphiques de groupe pratiques de nos jours [Ste98, DJ03, Pai99].

On dira qu'un système de chiffrement homomorphique d'anneaux est pratique si c'est un système de chiffrement homomorphique de groupes pratique, et si la taille des chiffrés augmente au plus linéairement quand on les multiplie entre eux.

A.2 Protocoles PIR homomorphiques à une dimension

Dans la construction proposée par Stern pour créer des protocoles PIR, l'idée de base est d'envoyer un message contenant plusieurs chiffrés de 0 et un seul chiffré de 1, ce dernier étant associé à l'élément de la base de données dans lequel l'utilisateur est intéressé. La base fera une multiplication scalaire du j-ème chiffré par le j-ème élément dans la base, puis fera la somme de tous les résultats. À cause des propriétés homomorphiques du système de chiffrement utilisé, le résultat des opérations de la base sera un chiffré de l'élément dans lequel l'utilisateur été intéressé.

Dans cette section nous commençons par définir formellement ce protocole puis nous donnons une preuve de qu'un tel protocole respecte les définitions données dans la section précédente.

Définition 8 *Soit* Π *un système de chiffrement homomorphique de groupes. Un protocole PIR homomorphique est une machine de Turing probabiliste à temps polynomial* Γ_n *qui pour une entrée k donne en sortie trois algorithmes* $Q(\cdot)$, $R(\cdot,\cdot)$, $X(\cdot,\cdot)$ *tels que,*

- $Q(\cdot)$ *est un algorithme probabiliste à temps polynomial qui pour une entrée* $i \in \{1,...,n\}$, *donne en sortie* $(c_1,...,c_n)$ *avec,* $c_i \leftarrow E(1)$, $c_j \leftarrow E(0)$ *pour* $j \in \{1,...,n\}$ $j \neq i$, *et* $(E,D) \leftarrow \Pi[k]$,
- $R(\cdot,\cdot)$ *est un algorithme déterministe à temps polynomial qui pour des entrées* $(c_1,...,c_n)$ *et* $\{e_1,...,e_n\}$, *donne en sortie* $\Sigma_{j\in\{1,...,n\}} e_j \cdot_{C_E} c_j$,
- $X(\cdot,\cdot)$ *est un algorithme déterministe à temps polynomial qui prend en entrée un chiffré r et un algorithme de déchiffrement D, et donne en sortie* $D(r)$.

Le résultat donné par l'algorithme d'extraction est correct tant que les éléments de la base de données ne dépassent pas l'ordre de l'élément 1 dans le groupe des textes clairs, à cause des propriétés du système de chiffrement utilisé. D'un autre côté, le secret du choix de l'utilisateur dépend des propriétés de sécurité du système de chiffrement. Le théorème ci-dessus prouve que si le protocole PIR n'est pas calculatoirement privé alors le système de chiffrement utilisé n'est pas sémantiquement sûr. On en déduit que tout PIR homomorphique qui utilise un système de chiffrement homomorphique sémantiquement sûr est calculatoirement privé.

Theorème 2 *Tout PIR homomorphique basé sur un système de chiffrement homomorphique sémantiquement sûr est calculatoirement privé dans le sens de la définition 5.*

Preuve :

Tout au long de cette preuve nous utiliserons les notations introduites dans la définition 8. Pour une requête $(c_1, ..., c_n) \leftarrow Q(i)$, et un contenu donné pour une base de données $\{e_1, ..., e_n\}$ on a,

$$D(\Sigma_{j\in\{1,...,n\}} e_j \cdot_{C_E} c_j) = \Sigma_{j\in\{1,...,n\}} e_j \cdot_{M_E} D(c_j) = \Sigma_{j\neq i} e_j \cdot_{M_E} 0 + e_i \cdot_{M_E} 1 = e_i$$

et donc le protocole vérifie la première propriété dans la définition 5.

Pour la seconde propriété, nous montrons que si elle n'est pas respectée alors le système de chiffrement n'est pas sémantiquement sûr. Supposons qu'il y ait un $c > 0$, un sous-ensemble non-fini $N' \subseteq \mathbb{N}$, une famille de circuits probabilistes à temps polynomial $C = \{C_k(\cdot,\cdot)\}$, et une distribution de probabilités P sur $\{1, ..., n\}$, tels que pour tout $k \in N'$,

$$Pr(C_k(\alpha, \mathcal{P}(k)) = i | i \overset{P}{\leftarrow} \{1, ..., n\}; Q \leftarrow \Gamma_n[k]; \alpha \leftarrow Q(i)) > p_M + k^{-c}$$

Montrons qu'il y a une famille de circuits probabilistes de taille polynomiale $G = G_k(\cdot,\cdot)$ telle que la sécurité sémantique de Π ne soit pas vérifiée pour la distribution uniforme sur $\{0,1\}$. Soit $k \in N'$. Prenons $(E, D) \leftarrow \Pi[k]$, $\alpha_1 \leftarrow E(1)$, $\alpha_2 \leftarrow E(0)$, et $\alpha \overset{U}{\leftarrow} \{\alpha_1, \alpha_2\}$. $G_k(\alpha, E)$ suit les étapes indiquées ci-dessous,

– $i \overset{P}{\leftarrow} \{1, ..., n\}$,
– définir $c_i = \alpha$, $c_j \leftarrow E(0)$ pour $j \neq i$,
– appeler $C_k((c_1, ..., c_n), \mathcal{P}(k)) = r$
– si $r = i$ donner en sortie 1, sinon donner en sortie 0.

Si $\alpha \in E(1)$, étant donné que

$$Pr(C_k(\alpha, \mathcal{P}(k)) = i | i \overset{P}{\leftarrow} \{1, ..., n\}; Q \leftarrow \Gamma_n[k]; \alpha \leftarrow Q(i)) > p_M + k^{-c}$$

on aura $Pr(G_k(\alpha, E) = 1) > p_M + k^{-c}$.

Si $\alpha \in E(0)$, on aura pour tout $j \in \{1, ..., n\}$, $c_j \in E(0)$. Comme tous les c_j ont été tirés en suivant la même distribution de probabilités (celle qui est associé à l'espace de probabilités $E(0)$), il n'y a pas d'information sur l'index i et donc la seule stratégie que C_k peut suivre est de deviner i. Dans tous les cas, comme i a été tiré de $\{1, ..., n\}$ en suivant la distribution de probabilités P on aura, $Pr(C_k((c_1, ..., c_n), \mathcal{P}(k)) = i) \leq p_M$ et donc $Pr(G_k(\alpha, E) = 0) \geq 1 - p_M$.

Enfin, comme $Pr(\alpha \in E(0)) = Pr(\alpha \in E(1)) = 1/2$, on a,

$$\begin{aligned} Pr(G_k(\alpha, E) = m | m \overset{U}{\leftarrow} \{0,1\}; (E, D) \leftarrow \Pi[k]; \alpha \leftarrow E(m)) \\ \geq 1/2(p_M + k^{-c}) + 1/2(1 - p_M) \geq 1/2 + k^{-c}/2 \end{aligned}$$

et donc Π n'est pas sémantiquement sûr.

□

A.3 Protocoles PIR homomorphiques à plusieurs dimensions

Dans la construction à une dimension, la requête d'un utilisateur est une chaîne de chiffrés. Celle-ci peut être utilisée sur une base de données à une dimension, ou sur une base de données représentée sur plusieurs dimensions. Dans ce dernier cas, la base utilise la méthode de Kushilevitz et Ostrovsky qui consiste à appliquer itérativement cette même requête sur plusieurs segments de données obtenant un résultat par segment (par exemple une fois par ligne d'une matrice, voir figure 3.2). Un segment de données donnant un unique résultat, l'ensemble des résultats peuvent être représentés sous une dimension de moins que la base (par exemple la matrice de bits devient une colonne de résidus). Cette méthode peut être appliquée récursivement afin de continuer à réduire les dimension (par exemple, une colonne de résidus devenant un seul et unique groupe de résidus), mais dans tous les cas les requêtes sont appliquées itérativement, en transformant des segments de blocs de données (dans le cas de Kushilevitz et Ostrovsky ces blocs étant de taille un) en un résultat unique.

Dans le cas à une dimension, une requête composée de $O(s)$ éléments permet de réduire un segment de s blocs de données en un unique résultat. Les protocoles PIR homomorphiques à plusieurs dimensions visent à ce qu'une requête composée de $O(s)$ éléments permette de réduire un hyper-cube de L dimensions et contenant s^L blocs de données, en un unique résultat ayant un facteur d'expansion comparable à celui qu'on obtient pour les protocoles à une dimension.

Étant donné que, dans tous les protocoles PIR connus, la réponse de la base de données a un facteur d'expansion strictement supérieur à un, l'utilisation récursive du protocole comme proposé par Kushilevitz et Ostrovsky donne lieu à une croissance géométrique [KO97, Ste98, GR05] ou linéaire [Lip05] de la réponse. Les protocoles PIR homomorphiques à plusieurs dimensions permettent de réduire la taille des requêtes sans utiliser la récursion, et constituent donc une alternative très intéressante.

A.3.1 Cas à deux dimensions

La construction de protocoles PIR homomorphiques à deux dimensions est ici décrite informellement pour aider le lecteur à comprendre le cas général.

Supposons que $n = s^2$ pour un entier s donné. Représentons les contenus de la base de données comme une matrice de taille $s \times s$. Avec cette représentation, les éléments de la base sont indexés par deux coordonnées, et on les notera $e_{(j_1,j_2)}$ avec $(j_1, j_2) \in \{1, ..., s\}^2$. Pour tout, $i \in \{1, .., n\}$ on associe un ensemble de deux index (i_1, i_2). Quand un utilisateur veut obtenir un élément $e_{(i_1,i_2)}$, il envoie deux ensembles de s chiffrés. Dans le premier ensemble, seulement le i_1-ème élément sera un chiffré de 1, le reste étant des chiffrés de 0. Dans le deuxième ensemble, le seul chiffré de 1 sera le i_2-ème élément. Notons $(c_{1,1}, ..., c_{1,s})$ le premier ensemble de chiffrés, et $(c_{2,1}, ..., c_{2,s})$ le deuxième ensemble. Pour chaque élément $e_{(j_1,j_2)}$ la base calcule $e_{(j_1,j_2)} \cdot_{C_E} c_{1,j_1} \times_{C_E} c_{2,j_2}$, réalise l'addition de tous ces résultats $R = \Sigma_{(j_1,j_2)\in\{1,...,s\}^2} e_{(j_1,j_2)} \cdot_{C_E} c_{1,j_1} \times_{C_E} c_{2,j_2}$, et envoie R à l'utilisateur.

L'utilisateur reçoit R, et si le système de chiffrement utilisé est un système de chiffrement homomorphique d'anneaux, lorsqu'il déchiffre R il obtient $e_{(i_1,i_2)}$ puisque pour tout $(j_1, j_2) \neq (i_1, i_2)$ on a $D(c_{1,j_1} \times_{C_E} c_{2,j_2}) = 0$, et $D(c_{1,i_1} \times_{C_E} c_{2,i_2}) = 1$.

Avec un tel protocole, l'utilisateur envoie $2 \times s$ chiffrés, et chaque requête permet d'extraire un élément parmi s^2, ce qui est très intéressant comparé aux protocoles à une dimension où une requête de s chiffrés permet d'extraire un élément parmi s uniquement. Dans la section suivante nous présentons formellement les protocoles PIR homomorphiques à plusieurs dimensions et nous prouvons leur sécurité. Cependant, il n'y a à notre connaissance qu'un seul système de chiffrement permettant d'implémenter de tels protocoles [BGN05]. De plus, ce système ne permet pas de travailler à plus de deux dimensions et se base sur une nouvelle supposition pour laquelle il est encore trop tôt pour décider quels devraient être les paramètres de sécurité et à quelles performances on peut s'attendre dans le facteur d'expansion. Ces résultats n'ont donc pour le moment, qu'un intérêt théorique, en montrant que des systèmes de chiffrement homomorphiques d'anneaux efficaces devraient permettre de réaliser des protocoles PIR très intéressants.

A.3.2 Cas général

Supposons que $n = s^d$ pour deux entiers donnés s et d. Afin de simplifier les notations on définit $I_s = \{1, ..., s\}$ et pour tout $j \in \{1, ..., n\}$ on associe un vecteur $\mathbf{j} = (j_1, ..., j_d) \in I_s^d$, pour lequel on notera $j \equiv (j_1, ..., j_d)$.

Définition 9 *Soit* Π *un système de chiffrement homomorphique d'anneaux. Un protocole PIR homomorphique à plusieurs dimensions est une machine de Turing probabiliste à temps polynomial* $\Gamma_{n,d}$ *qui pour une entrée k, donne en sortie trois algorithmes* $Q(\cdot)$, $R(\cdot,\cdot)$, $X(\cdot,\cdot)$ *tels que,*

- $Q(\cdot)$ *est un algorithme probabiliste à temps polynomial prenant en entrée* $i \in \{1,...,n\}$, $i \equiv (i_1,...,i_d)$, *et donnant en sortie pour tout* $l \in \{1,...,d\}$, *un ensemble* $(c_{l,1},...,c_{l,s})$ *avec,* $c_{l,i_l} \leftarrow E(1)$, $c_{l,j_l} \leftarrow E(0)$ *pour* $j_l \in I_s - \{i_l\}$, *et* $(E,D) \leftarrow \Pi[k]$,
- $R(\cdot,\cdot)$ *est un algorithme déterministe à temps polynomial qui pour des entrées* $((c_{1,1},...,c_{1,s}),...,(c_{d,1},...,c_{d,s}))$ *et* $\{e_{\mathbf{j}}|\mathbf{j} \in I_s^d\}$, *donne en sortie* $\Sigma_{\mathbf{j}\in I_s^d}\, e_{\mathbf{j}} \cdot_{C_E} c_{1,j_1} \times_{C_E} \cdots \times_{C_E} c_{d,j_d}$,
- $X(\cdot,\cdot)$ *est un algorithme déterministe à temps polynomial qui prend en entrée un chiffré r et un algorithme de déchiffrement D, et donne en sortie* $D(r)$.

Comme pour les PIR homomorphiques à une dimension, le résultat est correct à cause des propriétés homomorphiques du système de chiffrement utilisé, et le secret du choix de l'utilisateur dérive des propriétés de sécurité du système de chiffrement. Encore une fois, on prouve que si le secret du choix de l'utilisateur n'est pas respecté alors le système de chiffrement ne peut pas être sémantiquement sûr.

Theorème 3 *Tout protocole PIR homomorphique à plusieurs dimensions basé sur un système de chiffrement sémantiquement sûr est un protocole PIR calculatoirement privé dans le sens de la définition 5.*

Preuve :

En utilisant les notations données dans la définition 9 on sait que

$$D(\Sigma_{\mathbf{j}\in I_s^d}\, e_{\mathbf{j}} \cdot_{C_E} c_{1,j_1} \times_{C_E} \cdots \times_{C_E} c_{d,j_d}) = \Sigma_{\mathbf{j}\in I_s^d}\, e_{\mathbf{j}} \cdot_{M_E} D(c_{1,j_1}) \times_{M_E} \cdots \times_{M_E} D(c_{d,j_d})$$

De plus comme $D(c_{l,j_l})$ est égal à un si $j_l = i_l$ et à zéro sinon, on a

$$\Sigma_{\mathbf{j}\in I_s^d}\, e_{\mathbf{j}} \cdot_{M_E} D(c_{1,j_1}) \times_{M_E} \cdots \times_{M_E} D(c_{d,j_d}) = \Sigma_{\mathbf{j}\neq(i_1,...,i_d)}\, e_{\mathbf{j}} \cdot_{M_E} 0 + e_{(i_1,...,i_d)} \cdot_{M_E} 1 = e_{(i_1,...,i_d)}$$

et donc le protocole respecte la première propriété de la définition 5.

Pour la seconde propriété nous montrons encore une fois que si elle n'est pas respectée alors le système de chiffrement n'est pas sémantiquement sûr. Supposons qu'il ait un $c > 0$, un sous-ensemble non-fini $N' \subseteq \mathbb{N}$, une famille de circuits probabilistes de taille polynomiale $C = \{C_k(\cdot,\cdot)\}$, et une distribution de probabilités P sur $\{1,...,n\}$, tels que pour tout $k \in N'$,

$$Pr(C_k(\alpha, \mathcal{P}(k)) = i | i \overset{P}{\leftarrow} \{1,...,n\}; Q \leftarrow \Gamma_n[k]; \alpha \leftarrow Q(i)) > p_M + k^{-c}$$

Montrons qu'il y a une famille de circuits probabiliste de taille polynomiale $G = G_k(\cdot,\cdot)$ telle que la sécurité sémantique de Π ne soit pas vérifiée pour la distribution uniforme sur $\{0,1\}$. Soit $k \in N'$. On prend $(E,D) \leftarrow \Pi[k]$, $\alpha_1 \leftarrow E(1)$, $\alpha_2 \leftarrow E(0)$, et $\alpha \overset{U}{\leftarrow} \{\alpha_1, \alpha_2\}$. $G_k(\alpha, E)$ suit les étapes suivantes,

– $i \overset{P}{\leftarrow} \{1, ..., n\}$,
– définir $c_i = \alpha$, $c_j \leftarrow E(0)$ for $j \neq i$,
– appeler $C_k((c_1, ..., c_n), \mathcal{P}(k)) = r$
– si $r = i$ donner en sortie 1, sinon 0.

Si $\alpha \in E(1)$, étant donné que

$$Pr(C_k(\alpha, \mathcal{P}(k)) = i | i \overset{P}{\leftarrow} \{1, ..., n\}; Q \leftarrow \Gamma_n[k]; \alpha \leftarrow Q(i)) > p_M + k^{-c}$$

on aura $Pr(G_k(\alpha, E) = 1) > p_M + k^{-c}$.

Si $\alpha \in E(0)$, on aura pour tout $j \in \{1, ..., n\}$ $c_j \in E(0)$. Étant donné que tous les c_j ont étés tirés en suivant la même distribution de probabilités (celle qui est associée à l'espace de probabilités $E(0)$), il n'y a pas d'information sur la position de i et donc la seule stratégie que C_k peut suivre est de deviner l'index i. Dans tous les cas, puisque i a été tiré aléatoirement de $\{1, ..., n\}$ en suivant la distribution de probabilités P on a, $Pr(C_k((c_1, ..., c_n), \mathcal{P}(k)) = i) \leq p_M$ et donc $Pr(G_k(\alpha, E) = 0) \geq 1 - p_M$.

Enfin, comme $Pr(\alpha \in E(0)) = Pr(\alpha \in E(1)) = 1/2$, on a,

$$\begin{aligned} Pr(G_k(\alpha, E) = m | m \overset{U}{\leftarrow} \{0,1\}; (E,D) \leftarrow \Pi[k]; \alpha \leftarrow E(m)) \\ \geq 1/2(p_M + k^{-c}) + 1/2(1 - p_M) \geq 1/2 + k^{-c}/2 \end{aligned}$$

et donc Π n'est pas sémantiquement sûr.

□

Le coût des communications d'un tel système dépendra du système de chiffrement. Si lors de la multiplication de deux chiffrés la taille du résultat augmente linéairement, les performances obtenues seront analogues à celles du protocole de Lipmaa, avec éventuellement des tailles de requête moindres. Si la multiplication de deux chiffrés ne donne pas lieu à une augmentation de la taille, comme c'est souvent le cas pour la somme, en représentant la base de données comme un $log(n)$-hyper-cube, les requêtes auront une taille qui grandira logarithmiquement en n et le facteur d'expansion de la réponse sera constant. Nous obtiendrons donc un protocole PIR optimal avec faible coût d'initialisation (uniquement l'échange des paramètres du système de chiffrement).

Bibliographie

[AD05] Carlos Aguilar Melchor and Yves Deswarte. pMIX : Untraceability for Small Hiding Groups. In *Fourth IEEE International Symposium on Network Computing and Applications*, pages 29–40, 2005.

[AD06] Carlos Aguilar Melchor and Yves Deswarte. From DC-nets to pMIXes : multiple variants for anonymous communications. In *Fifth IEEE International Symposium on Network Computing and Applications*, pages 163–172, 2006.

[ADS03] Alessandro Acquisti, Roger Dingledine, and Paul F. Syverson. On the Economics of Anonymity. In *Financial Cryptography*, pages 84–102, 2003.

[Agu06a] Carlos Aguilar Melchor. An overview of single-database private information retrieval schemes and their performances. Technical report, Rapport 06149, LAAS-CNRS, 2006.

[Agu06b] Carlos Aguilar Melchor. Anonymous Communication on Local Area Networks. Technical report, Rapport 06148, LAAS-CNRS, 2006.

[Amb97] Andris Ambainis. Upper Bound on Communication Complexity of Private Information Retrieval. In *ICALP*, pages 401–407, 1997.

[ANZ] The Anonymizer. `http ://www.anonymizer.com`.

[BdB90] Jurjen Bos and Bert den Boer. Detection of Disrupters in the DC Protocol. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology—EUROCRYPT 89*, volume 434 of *LNCS*, pages 320–328. Springer-Verlag, 1990.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF Formulas on Ciphertexts. In *TCC 2005*, volume 3378 of *LNCS*, pages 325–341, 2005.

[BIKR02] Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and Jean-François Raymond. Breaking the $O(n^{1/(2k-1)})$ Barrier for Information-Theoretic Private Information Retrieval. In *FOCS : IEEE Symposium on Foundations of Computer Science (FOCS)*, 2002.

[BMS01] Adam Back, Ulf Möller, and Anton Stiglic. Traffic Analysis Attacks and Trade-Offs in Anonymity Providing Systems. In Ira S. Moskowitz, editor, *Proceedings of Information Hiding Workshop (IH 2001)*, pages 245–257. Springer-Verlag, LNCS 2137, Avril 2001.

[BPS00] Oliver Berthold, Andreas Pfitzmann, and Ronny Standtke. The Disadvantages of Free MIX Routes and how to Overcome Them. In *Workshop on Design Issues in Anonymity and Unobservability*, pages 30–45, 2000.

[BY87] Ernest F. Brickell and Yacov Yacobi. On Privacy Homomorphisms (Extended Abstract). In David Chaum and Wyn L. Price, editors, *Advances in Cryptology—EUROCRYPT 87*, volume 304 of *LNCS*, pages 117–125. Springer-Verlag, 1987.

[CB95] David A. Cooper and Kenneth Birman. Preserving Privacy in a Network of Mobile computers. In *IEEE Symposium on Security and Privacy*, pages 26–38, May 1995. http ://www.cs.cornell.edu/Info/People/dcooper/dcooper.html.

[CG97] Benny Chor and Niv Gilboa. Computationally Private Information Retrieval (Extended Abstract). In *STOC*, pages 304–313, 1997.

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private Information Retrieval. In *FOCS : IEEE Symposium on Foundations of Computer Science (FOCS)*, 1995.

[Cha81] David Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications ACM*, 24(2) :84–88, 1981.

[Cha85] David Chaum. Security Without Identification : Transaction Systems to Make Big Brother Obsolete. *Communications of the ACM*, 28(10) :1030–1044, October 1985.

[Cha88] David Chaum. The Dining Cryptographers Problem : Unconditional Sender and Recipient Untraceability. *J. Cryptology*, 1(1) :65–75, 1988.

[Cha04] Yan-Cheng Chang. Single Database Private Information Retrieval with Logarithmic Communication. In *ACISP : Information Security and Privacy : Australasian Conference*, 2004.

[CMS99] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally Private Information Retrieval with Polylogarithmic Communication. In *EUROCRYPT : Advances in Cryptology : Proceedings of EUROCRYPT*, 1999.

[DA06a] Yves Deswarte and Carlos Aguilar Melchor. Current and Future Privacy Enhancing Technologies for the Internet. *Annals of Telecommunications / Annales des Télécommunications*, Volume 61(3-4), Lavoisier, pages 399–417, 2006.

[DA06b] Yves Deswarte and Carlos Aguilar Melchor. Protection de la vie privée sur Internet. *Revue de l'Électricité et de l'Électronique (REE)* à paraître en 2006, Hermès-Science, 2006.

[DA06c] Yves Deswarte and Carlos Aguilar Melchor. *Technologies de protection de la vie privée sur Internet*, dans *Sécurité des systèmes d'information*, pages 49–72. Réseaux et Télécoms. Hermès-Science, 2006.

[Dan00] George Danezis. The Implementation of an Auction Protocol over Anonymous Networks. `http ://citeseer.ist.psu.edu/531250.html`, 2000.

[Dan04] George Danezis. The Traffic Analysis of Continuous-Time Mixes. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, volume 3424 of *LNCS*, Mai 2004.

[DDM03] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion : Design of a Type III Anonymous Remailer Protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, Mai 2003.

[DF96] Josep Domingo-Ferrer. A new privacy homomorphism and applications. *Inf. Process. Lett*, 60(5) :277–282, 1996.

[DH76] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6) :644–652, Nov 1976.

[DJ03] Ivan Damgård and Mads Jurik. A Length-Flexible Threshold Cryptosystem with Applications. In *ACISP 2003*, pages 350–364, 2003.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor : The Second-Generation Onion Router. In *USENIX Security Symposium*, pages 303–320. USENIX, 2004.

[DO00] Shlomi Dolev and Rafail Ostrovsky. Xor-Trees for Efficient Anonymous Multicast and Reception. *j-TISSEC*, 3(2) :63–84, May 2000.

[Dou02] John Douceur. The Sybil Attack. In *Proceedings of the 1st International Peer To Peer Systems Workshop (IPTPS 2002)*, Mars 2002.

[DP04] Claudia Díaz and Bart Preneel. Taxonomy of Mixes and Dummy Traffic. In *International Information Security Workshops*, pages 215–230, 2004.

[DS03] Claudia Díaz and Andrei Serjantov. Generalising Mixes. In Roger Dingledine, editor, *Proceedings of Privacy Enhancing Technologies workshop (PET 2003)*. Springer-Verlag, LNCS 2760, Mars 2003.

[DSDa] JAP, Anonymity and Privacy. `http ://anon.inf.tu-dresden.de`.

[DSDb] German police intrusion in the JAP system. `http ://www.datenschutzzentrum.de/material/themen/presse/anon-bka_e.htm`.

[FM02] Michael J. Freedman and Robert Morris. Tarzan : a Peer-to-Peer Anonymizing Network Layer. In *ACM Conference on Computer and Communications Security (CCS 2002)*, pages 193–206, 2002.

[Gee05] David Geer. Malicious Bots Threaten Network Security. *IEEE Computer*, 38(1) :18–20, 2005.

[GJ04a] Philippe Golle and Ari Juels. Dining Cryptographers Revisited. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 456–473. Springer, 2004.

[GJ04b] Philippe Golle and Ari Juels. Parallel Mixing. In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS 2004)*. ACM Press, Octobre 2004.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2) :270–299, Avril 1984.

[GR05] Craig Gentry and Zulfikar Ramzan. Single-Database Private Information Retrieval with Constant Communication Rate. In *ICALP : Annual International Colloquium on Automata, Languages and Programming*, 2005.

[GRS99] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Onion Routing. *Communications ACM*, 42(2) :39–41, 1999.

[GT96] Ceki Gülcü and Gene Tsudik. Mixing E-mail with Babel. In *Proceedings of the Network and Distributed Security Symposium - NDSS '96*, pages 2–16. IEEE, Fevrier 1996.

[Hel95] Johan (Julf) Helsingius. The Penet Anonymous Server. `http ://anon.penet.fi`, 1995.

[Jak99] Markus Jakobsson. Flash Mixing. In *Proceedings of Principles of Distributed Computing - PODC '99*. ACM Press, 1999.

[Jea06] Jean-Pierre Soulès. Téléphonie : pourquoi n'externalisez-vous pas ?. *01 Informatique*, 1858 :38–41, mai 2006.

[JMP+98] Anja Jerichow, Jan Müller, Andreas Pfitzmann, Birgit Pfitzmann, and Michael Waidner. Real-Time MIXes : A Bandwidth-Efficient Anonymity Protocol. *IEEE Journal on Selected Areas in Communications*, 16(4), 1998.

[KEB98] Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-and-Go MIXes : Providing Probabilistic Anonymity in an Open System. In *Proceedings of Information Hiding Workshop (IH 1998)*. Springer-Verlag, LNCS 1525, 1998.

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication Is Not Needed : Single Database, Computationally-Private Information Retrieval (extended abstract). In *FOCS : IEEE Symposium on Foundations of Computer Science (FOCS)*, 1997.

[Lip05] Helger Lipmaa. An Oblivious Transfer Protocol with Log-Squared Communication. In Jianying Zhou, Javier Lopez, Robert H. Deng, and Feng Bao, editors, *ISC*, volume 3650 of *LNCS*, pages 314–328. Springer, 2005.

[LS02] Brian Neil Levine and Clay Shields. Hordes : a Multicast-Based Protocol for Anonymity. *Journal of Computer Security*, 10(3) :213–240, 2002.

[Man04] Eran Mann. Private Access to Distributed Information, Master's Thesis, January 05 2004.

[Mar99] David Michael Martin. *Local anonymity in the Internet*. PhD thesis, Boston University, 1999.

[MD05] Steven J. Murdoch and George Danezis. Low-Cost Traffic Analysis of Tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*. IEEE CS, Mai 2005.

[MK00] Masashi Mitomo and Kaoru Kurosawa. Attack for Flash MIX. In *Proceedings of ASIACRYPT 2000*. Springer-Verlag, LNCS 1976, 2000.

[MRS88] Silvio Micali, Charles Rackoff, and Bob Sloan. The Notion of Security for Probabilistic Cryptosystems. *SICOMP : SIAM Journal on Computing*, 17, 1988.

[NIS] NIST. Random Number Generation. `http ://csrc.nist.gov/CryptoToolkit/tkrng.html`.

[Pai99] Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *EUROCRYPT : Advances in Cryptology : Proceedings of EUROCRYPT*, 1999.

[Pfi85] Andreas Pfitzmann. How to Implement Isdns Without User Observability - some Remarks. Technical report, Karlsruhe, 1985.

[Pfi90] Andreas Pfitzmann. *Diensteintegrierende Kommunikationsnetze mit teilnehmerüberprüfbarem Datenschutz*, volume 234 of *Informatik-Fachberichte*. Springer, 1990.

[PK00] Andreas Pfitzmann and Marit Köhntopp. Anonymity, Unobservability, and Pseudonymity — A Proposal for Terminology. In Hannes Federrath, editor, *Designing Privacy Enhancing Technologies*, volume 2009 of *LNCS*, pages 1–9, Berkeley, CA, USA, July 2000. Springer-Verlag, Berlin Germany.

[PNT] The Church of Scientology vs. Penet. `http ://www.thecia2.net/users/rnewman/scientology/anon/penet.html`.

[PP90] Birgit Pfitzmann and Andreas Pfitzmann. How to Break the Direct RSA-Implementation of MIXes. In *Proceedings of EUROCRYPT 1989*. Springer-Verlag, LNCS 434, 1990.

[PPW91] Andreas Pfitzmann, Birgit Pfitzmann, and Michael Waidner. ISDN-mixes : Untraceable communication with very small bandwidth overhead. In *Proceedings of the GI/ITG Conference on Communication in Distributed Systems*, pages 451–463, Fevrier 1991.

[PW85] Andreas Pfitzmann and Michael Waidner. Networks without User Observability—Design Options. In Franz Pichler, editor, *Advances in Cryptology—EUROCRYPT 85*, volume 219 of *LNCS*, pages 245–253. Springer-Verlag, 1986, April 1985.

[Ray00] Jean-François Raymond. Traffic Analysis : Protocols, Attacks, Design Issues, and Open Problems. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies : Workshop on Design Issues in Anonymity and Unobservability*, pages 10–29. Springer-Verlag, LNCS 2009, Juillet 2000.

[RP04] Marc Rennhard and Bernhard Plattner. Practical Anonymity for the Masses with MorphMix. In *Financial Cryptography*, pages 233–250, 2004.

[RR98] Michael K. Reiter and Aviel D. Rubin. Crowds : Anonymity for Web Transactions. *ACM Trans. Inf. Syst. Secur.*, 1(1) :66–92, 1998.

[SBS02] Rob Sherwood, Bobby Bhattacharjee, and Aravind Srinivasan. P5 : A Protocol for Scalable Anonymous Communication. In *IEEE Symposium on Security and Privacy*, page 58, 2002.

[SG02] Rainer Steinwandt and Willi Geiselmann. Cryptanalysis of Polly Cracker. *IEEE Transactions on Information Theory*, 48(11) :2990–2991, 2002.

[Ste98] Julien P. Stern. A New Efficient All-Or-Nothing Disclosure of Secrets Protocol. In Kazuo Ohta and Dingyi Pei, editors, *ASIACRYPT*, volume 1514 of *LNCS*, pages 357–371. Springer, 1998.

[vABH03] Luis von Ahn, Andrew Bortz, and Nicholas J. Hopper. k-Anonymous Message Transmission. In *ACM Conference on Computer and Communications Security*, pages 122–130, 2003.

[Wai89] Michael Waidner. Unconditional Sender and Recipient Untraceability in Spite of Active Attacks. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology—EUROCRYPT 89*, volume 434 of *LNCS*, pages 302–319. Springer-Verlag, 1990, 10–13 April 1989.

[WP89] Michael Waidner and Birgit Pfitzmann. The Dining Cryptographers in the Disco : Unconditional Sender and Recipient Untraceability with Computationally Secure Serviceability. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology—EUROCRYPT 89*, volume 434 of *LNCS*, page 690. Springer-Verlag, 1990, 10–13 April 1989.

[YF04] Gang Yao and Dengguo Feng. A New k-Anonymous Message Transmission Protocol. In *WISA*, pages 388–399, 2004.

Thèse de doctorat de M. Aguilar Melchor "Les communications anonymes à faible latence"

RÉSUMÉ

Dans ce mémoire nous proposons des systèmes de communication anonyme à faible latence. Pour cela nous étudions les performances des systèmes basés sur les primitives classiques : envoi superposé ou bourrage chiffré pour l'émission, et diffusion avec adressage implicite pour la réception, quand les groupes d'utilisateurs potentiels sont contraints à être de petite taille, peu changeants, ou localisés. Nous proposons l'utilisation des protocoles de récupération d'informations privée (ou protocoles PIR pour Private Information Retrieval) comme alternative à la diffusion avec adressage implicite, et nous étudions les systèmes en résultant. Ces systèmes permettent de réduire significativement le coût des communications, au prix d'un coût calculatoire important. Au moyen d'exemples, nous montrons que ces nouvelles solutions offrent un meilleur choix dans certaines situations, notamment pour les utilisateurs connectés au service par Internet. Dans un deuxième temps, nous mettons en avant les relations entre les différentes techniques et montrons que les systèmes basés sur les primitives classiques ne sont en fait que des instances d'une famille qui, par l'utilisation de la récupération d'informations privée, devient nombreuse et polyvalente. Ainsi, on dispose de beaucoup plus de degrés de liberté pour l'implémentation de solutions fournissant des communications anonymes dans le cadre des groupes d'utilisateurs sous contraintes.

Mots-clés : protection de la vie privée, communications anonymes, récupération d'informations privée.

"Low-latency Anonymous Communications"

ABSTRACT

In this thesis, we present different systems providing low-latency anonymous communications. We first study the performance of systems based on well known primitives such as superposed sending and encrypted padding for transmission, and broadcast with implicit addressing for reception, when the group of potential users is restricted to be small, closed, or localized. We propose the usage of Private Information Retrieval (PIR) protocols as an alternative to broadcast with implicit addressing, and we study the resulting systems. These systems allow us to trade communication cost, for computational cost. Through some examples, we show that the new solutions offer a better choice in some situations, specially when the users are connected to the service through the Internet. Then, we put forward how the different approaches are related, and show that the systems based on classic primitives are in fact just some instances of a family, which becomes much larger and versatile through the introduction of PIR protocols.

Keywords : Privacy, anonymous communications, private information retrieval.